Automatic Generation of 3D Motion Based on Text Descriptions

Bachelor's Thesis

Bachelorarbeit für Informatik und Wirtschaftsinformatik

Eliyazar Ganchev

12026921

Advisor

Associate Prof. Dr.techn. Dipl.-Ing. Clemens Heitzinger

Vienna, 8. Oktober 2025

Contents

Abstract

Turning written descriptions into 3D animations is no longer a futuristic idea. Thanks to ongoing progress in machine learning and generative modeling, it's now possible to generate human motion directly from natural language. This thesis explores how full-body motion can be synthesized using text prompts and introduces a working prototype to demonstrate the idea.

The project takes inspiration from models like T2M-GPT but moves in a different direction by building a compact system from the ground up. The approach uses CLIP embeddings combined with a basic neural network to generate motion, which is then animated inside Blender. A user-friendly interface allows for simple text input and immediate playback of the resulting motion.

While the system is still in its early stages and hasn't been trained on larger datasets, it already provides a clear example of how textual input can be transformed into motion. The thesis also reviews evaluation methods and highlights possible paths for improving realism, motion structure, and control in future versions.

Keywords: Text-to-Motion Generation, 3D Motion Synthesis, Multimodal Representation Learning

CHAPTER 1

Introduction

The idea of controlling a 3D character with a simple sentence has been part of research discussions for a long time, especially in animation and artificial intelligence. Today, this concept is beginning to take real shape, thanks to developments in how computers process both language and movement. As more digital environments use animated avatars and virtual agents, language is becoming a practical way to drive motion. Instead of manually designing every action, users could just describe it in text and let a system generate the animation [1, 2].

This growing possibility is based on ideas from different fields. Language models are better at grasping meaning, and motion generators have become more flexible and expressive. Some newer systems map both text and motion into the same kind of vector space, which allows them to match one with the other more directly [3, 4]. Other approaches use techniques like masked modeling to improve structure, or diffusion processes to create smoother, more lifelike sequences [5].

My own project started with a simple question: how much of this could be done using a smaller, more understandable system? Rather than working with a large pretrained model, I decided to use CLIP to embed sentences and then pass those embeddings through a lightweight neural network to generate motion [6, 7].

Once generated, the motion is saved as a NumPy array and visualized in Blender. I also developed a Blender interface where users can type a sentence like "jump forward" and see the cube or skeleton animate accordingly. In the background, the system connects to several concepts from current research, especially in how motion and language are paired or disentangled [3, 8].

Even though the system hasn't been trained on real-world datasets yet, it already shows that basic motion can be produced from text. It provides a simple structure that can be expanded later with more training, skeletal complexity, and control options. While

still experimental, it shows that text-based motion generation is not only possible, but practical and worth improving [2, 6].

CHAPTER 2

Theoretical Background

2.1 Human Motion Representation

Human motion is typically represented through skeletal models, where each joint follows a hierarchical structure that reflects the anatomy of the human body. Each joint has either a position in 3D space or a rotational value that changes over time to form continuous animation. Most modern systems use a format where motion is stored as a tensor with the shape (frames, joints, 3), representing the position of each joint in every frame [2].

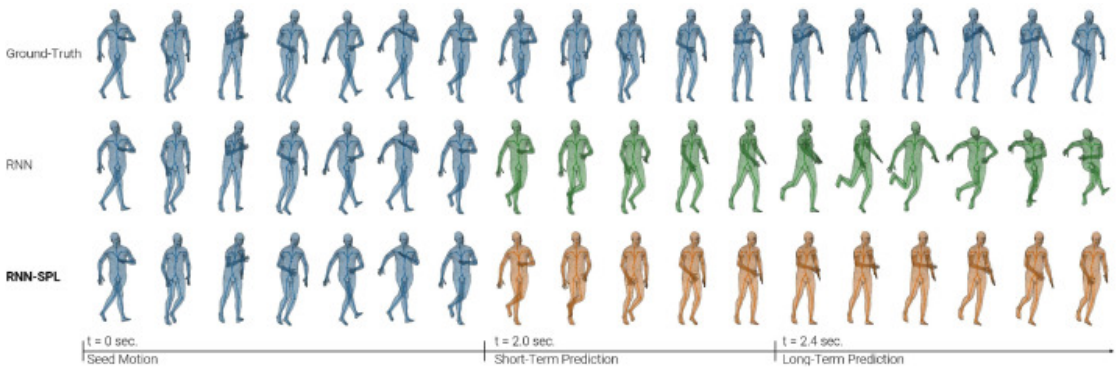

Figure 2.1: Example of human motion prediction using different models. The first row shows the ground-truth motion. The second and third rows show predictions generated by an RNN and a variant with structure-preserving loss (RNN-SPL). A seed motion segment is given as input (blue), followed by short-term (green) and long-term (orange) predictions [2].

The choice of data format depends on the task. In this work, motion is stored as `.npy` arrays due to their efficiency and compatibility with machine learning workflows. Other

widely used formats include `.bvh`, which stores both hierarchy and animation curves, and `.fbx`, which is often used in production pipelines for its compatibility with animation software like Blender.

Temporal smoothness is a key factor in whether a motion looks realistic. It is not enough for each pose to look correct; the transitions between frames must also follow natural rhythms of acceleration and deceleration [9]. When these transitions are abrupt or inconsistent, even a technically accurate pose sequence can appear artificial. Evaluating both the spatial accuracy and the temporal dynamics of motion is essential to producing lifelike results see Figure 2.1.

2.2 Natural Language and Semantic Understanding

Text commands provide an intuitive interface for describing motion, but natural language can be ambiguous and context-dependent. Phrases like "turn slightly" or "move slowly" contain implied timing and intensity that need to be interpreted correctly to produce meaningful animations. Extracting such information requires more than basic tokenization or syntactic parsing [6].

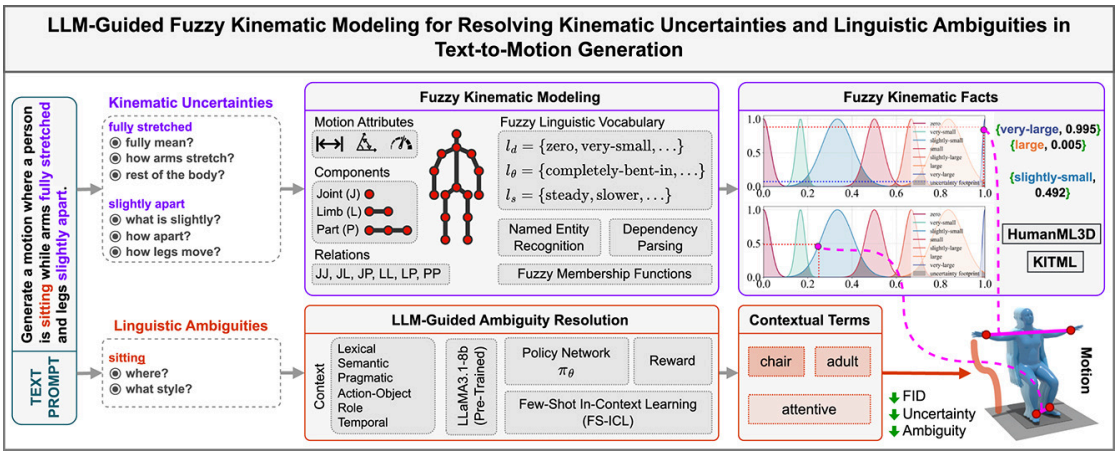

Figure 2.2: Overview of how LLM-guided fuzzy kinematic modeling resolves linguistic ambiguities and motion uncertainties. The system processes imprecise text prompts (e.g., "fully stretched", "slightly apart") by mapping them to fuzzy linguistic vocabularies, context-aware interpretations, and semantic role understanding. Final outputs are converted into motion sequences with reduced ambiguity [6].

To work with textual input, this project uses CLIP, a model that maps text into a high-dimensional embedding space shared with visual data. These embeddings are used to represent the semantic content of a sentence in a form suitable for neural networks. This approach enables the system to compare or align different types of input based on shared meaning, rather than direct word-to-motion mappings.

Other approaches address language ambiguity through text expansion and semantic enrichment. By generating multiple interpretations or adding contextual information to short prompts, models can better predict the intended movement [4]. These techniques help improve the quality of the generated motion without needing larger datasets or more complex architectures. A detailed overview of how fuzzy logic and large language models work together to resolve such ambiguities is shown in Figure 2.2.

2.3 Multimodal Representation Learning

Multimodal learning aims to connect different types of data, in this case, language and motion, by projecting them into a common latent space. In such a space, it becomes possible to generate motion from a sentence, retrieve a motion clip that matches a description, or evaluate the similarity between text and movement.

One method for learning this alignment is masked modeling, where a portion of the motion is hidden and predicted based on the corresponding text [3]. This encourages the model to form deeper associations between the two modalities. Another method is diffusion-based generation, where motion is produced by gradually refining random noise under the guidance of textual input [5].

Many models use simple projection formulas to align features across modalities. A standard approach is to apply a linear transformation, often expressed as

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

where $\mathbf{x}$ is either a text or motion embedding, and $\mathbf{W}$ and $\mathbf{b}$ are learnable parameters. These projections allow both types of input to coexist and interact in a shared representation space [3].

Although this thesis uses a simplified architecture, the core idea remains the same: embed the input sentence, map it to a shared space, and generate motion based on that representation. These principles form the conceptual foundation of many current systems in text-to-motion research.

CHAPTER 3

Related Work

3.1 Surveys and Taxonomies

A clear understanding of the landscape of human motion generation is essential for identifying both the strengths and gaps in current research. Comprehensive surveys offer structured overviews of available techniques and provide a foundation for comparing different approaches. These works often group models based on input modality, architectural choice, training objectives, and evaluation strategy. They also outline the evolution of the field, showing how early rule-based or motion-capture-based systems have gradually been replaced by neural networks capable of learning motion patterns from large datasets.

Recent surveys have placed particular emphasis on the shift from pose-level synthesis to semantically meaningful motion generation. They highlight how modern systems attempt to preserve motion realism while enabling high-level control, such as text or emotion conditioning. These overviews also point to the growing need for unified evaluation benchmarks and standardized motion datasets [1, 2].

3.2 Disentangled and Controllable Representations

Controllability is one of the key challenges in human motion generation. While many systems can produce plausible motion sequences, few offer meaningful control over what is generated. This is especially limiting when the user wants to manipulate specific properties like gait speed, emotional tone, or limb positioning.

Disentangled representation learning aims to address this limitation by separating the motion space into interpretable components. These components can be modified independently, allowing for flexible yet structured control. Some systems achieve this by designing the latent space so that each axis or cluster corresponds to a specific factor

of variation. Others introduce conditioning variables that guide the network toward different styles or attributes during generation. This separation enables reusability and makes it easier to experiment with hybrid or blended motions [7] see Figure 3.1.

3.3 Diffusion and Multimodal Models

Diffusion-based generative models have emerged as a powerful alternative to traditional autoregressive or GAN-based methods. They work by starting with random noise and gradually refining it into coherent outputs over several steps. This process has been particularly successful in motion generation, as it naturally supports smooth transitions and long-range temporal consistency.

In multimodal settings, diffusion models are often conditioned on external inputs such as text or audio. The goal is to align semantic signals with motion features at each stage of the generation. These systems not only improve motion quality but also enable synchronization with external modalities, such as speech rhythm or gesture intent.

By combining diffusion sampling with textual conditioning, models can create motion that is not only visually smooth but also semantically accurate. This makes diffusion particularly well-suited for co-speech gesture generation and text-based action synthesis [5, 8].

3.4 Text-Motion Retrieval and Semantic Expansion

Text-to-motion generation does not always require synthesizing new sequences from scratch. Retrieval-based methods offer a simpler and often more efficient alternative. These systems work by mapping both motion and text into a shared embedding space, where similarity scores are used to find the best-matching motion for a given prompt.

One of the main advantages of retrieval is that it can draw from high-quality, human-annotated motion datasets without the risk of producing unrealistic or unstable outputs. However, retrieval models depend heavily on the quality of text-motion alignment. To improve this, some systems use semantic expansion techniques that enrich sparse prompts with synonyms, contextual cues, or inferred action descriptors. This helps match vague user queries to more specific motion entries in the database.

Other models use masked modeling approaches to train the system on partially obscured motion fragments and text inputs. This allows them to learn finer-grained associations between individual motion features and their semantic counterparts [4, 3].

3.5 LLM-Guided Modeling

Natural language can be vague, subjective, or context-sensitive, which makes precise motion generation challenging. Large language models (LLMs) are increasingly being

used to bridge this gap by interpreting textual ambiguity and turning it into structured, actionable input for motion systems.

Instead of treating motion generation as a direct mapping from text to movement, some models treat language as a reasoning layer that guides fuzzy interpretation. In this context, "fuzzy" refers to linguistic terms like "slightly," "almost," or "strongly," which don not correspond to exact numerical values but instead to semantic ranges. By integrating LLMs with fuzzy logic systems, it becomes possible to resolve these uncertainties and produce more grounded motion output.

This strategy is especially valuable when working with real users who may not phrase their prompts in precise, technical terms. It also allows models to generalize better across different phrasing styles, regional expressions, and user habits [6].

3.6 Evaluation Techniques

Quantitative evaluation of motion generation models remains an open and debated issue. While some metrics like Fréchet Inception Distance (FID) or diversity scores are widely used, they often fail to capture perceptual quality or semantic accuracy, see Figure 3.2. Others rely on text-motion embedding distances, which correlate better with user intent but may still lack robustness.

Many researchers now advocate for hybrid evaluation strategies that combine automatic metrics with human judgment. For example, user studies that ask participants to rate realism, consistency, or faithfulness to the text description provide valuable insights that numerical metrics alone cannot capture.

There is also a growing recognition that motion quality should be evaluated across multiple dimensions: realism, diversity, semantic alignment, and controllability. A consistent framework for such evaluations is essential for comparing results across models, especially as architectures and datasets continue to evolve [9].

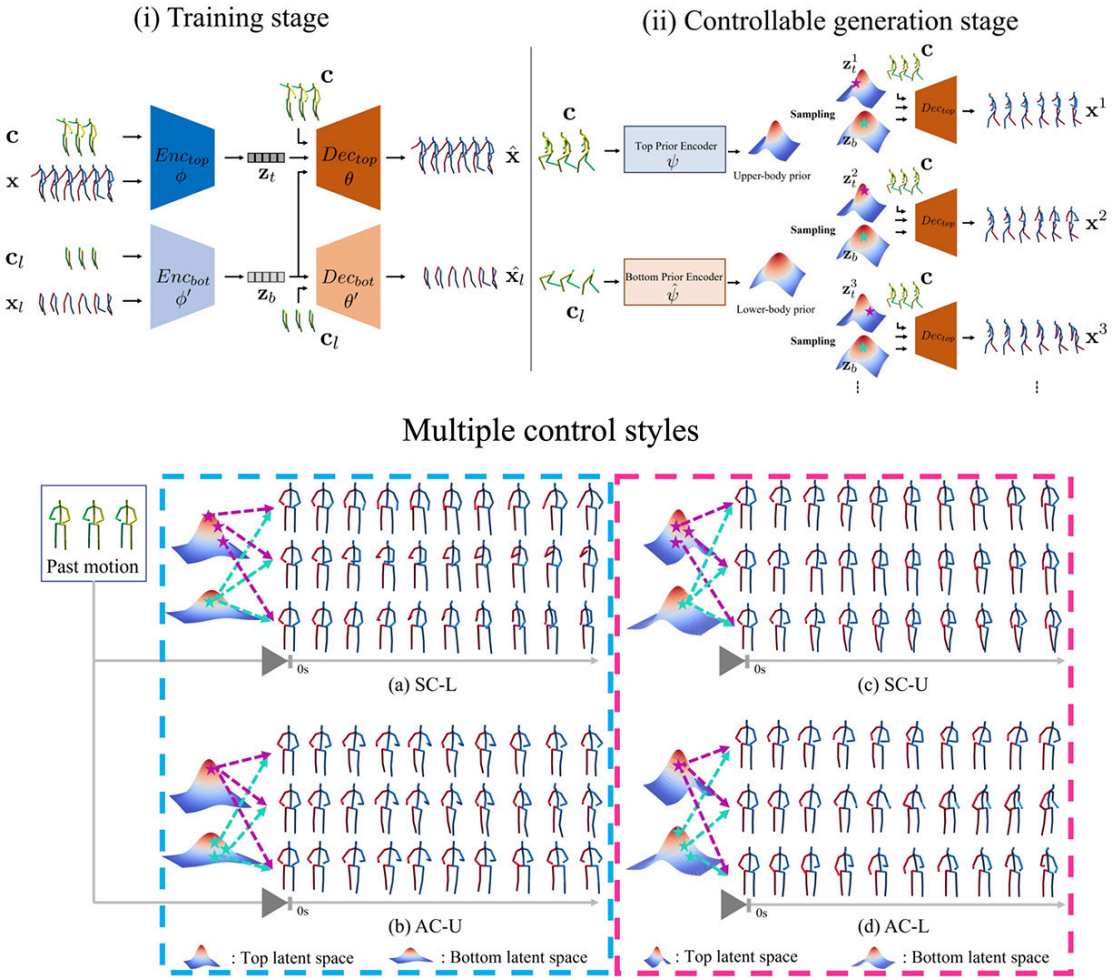

Figure 3.1: Overview of a disentangled motion generation model that separates motion into hierarchical latent spaces. The model supports various control styles by isolating motion attributes across different body parts and latent levels. This allows for modular editing and controllable synthesis [7].

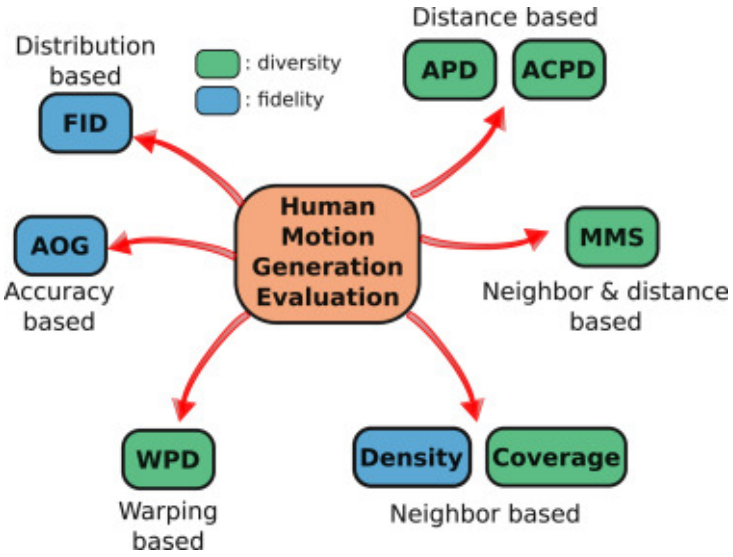

Figure 3.2: Classification of human motion generation evaluation metrics by type and measurement focus. Metrics include fidelity-based scores such as FID, AOG, and WPD, as well as diversity or coverage-based scores like APD, ACPD, MMS, and Density [9].

CHAPTER 4

Implementation and Pipeline

“

4.1 Overview of T2M-GPT

T2M-GPT is a transformer-based generative model designed for text-conditioned human motion synthesis. It operates by mapping text descriptions and motion sequences into a shared latent space using a vector quantized variational autoencoder (VQ-VAE). The core idea behind T2M-GPT is to discretize motion data into tokenized embeddings, allowing the model to treat motion generation as a language modeling task over motion tokens [1, 4].

The model is trained on datasets such as HumanML3D and KIT-ML, which pair natural language prompts with 3D motion clips. Each motion is encoded into a series of codebook indices representing compressed movement features. The text prompt is processed using CLIP, and the combined embedding is passed through a transformer decoder that generates motion tokens autoregressively. These tokens are later decoded back into full-body motion sequences.

The output format typically represents joint positions in 3D space across time, using a tensor of shape (T, J, 3), where T is the number of frames and J the number of joints. The original model supports skeletons with up to 22 joints and outputs NumPy arrays that can be visualized in animation tools like Blender.

For this thesis, the T2M-GPT model was installed and executed locally using its official open-source codebase. The prompt “The character jumps” was selected to evaluate the model’s ability to capture a dynamic, vertical motion. The resulting motion sequence was exported as a `.npy` file and visualized in Blender using a custom Python script. Four representative frames were extracted to illustrate the key stages of the motion, including standing, takeoff, mid-air, and landing. All outputs, visualizations, and observations

presented in this section are based on my own experimentation with the model. An example sequence is shown in Figure 4.1, highlighting the model's temporal coherence and alignment with the input description.

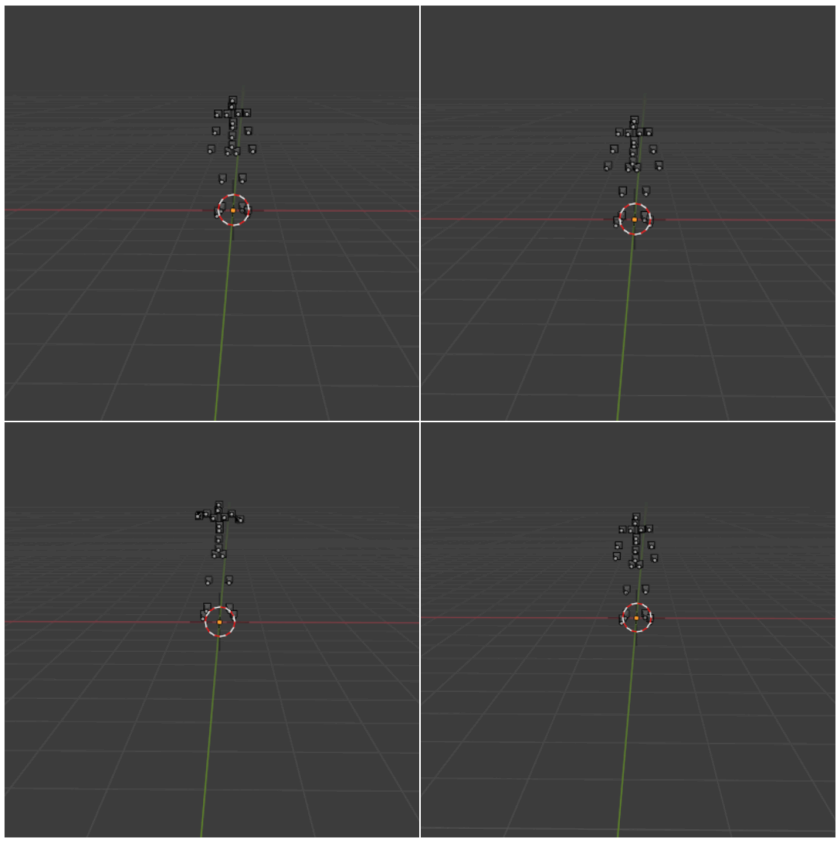

Figure 4.1: Collage of four keyframes from a motion sequence generated using T2M-GPT in response to the prompt "The character jumps." The frames represent the standing pose (top-left), takeoff (top-right), mid-air phase (bottom-left), and landing preparation (bottom-right) (screenshot from author's Blender setup).

4.2 Prototype Animation in Blender

Before integrating deep learning models, a simple rule-based animation prototype was developed in Blender to test the animation pipeline. This version uses hardcoded verbs to control the motion of a cube object. The verbs "walk," "jump," and "wave" are mapped to corresponding transformations in space or rotation.

The animation logic is written in Python using Blender's scripting API. A predefined list of action verbs is iterated over, and at each step, the cube's position or orientation is modified and a keyframe is inserted.

Algorithm 4.1: Rule-based Cube Animation (Implementation by Author)

Input: List of action verbs $V = \{$"walk", "jump", "wave"$\}$
Output: Animated cube in Blender

```
Initialize Blender scene;
Add cube at initial position;
foreach verb v in V do
    Insert keyframe at current frame;
    if v is "walk" then
        Move cube along x-axis;
    else
        if v is "jump" then
            Move cube upward along z-axis;
        else
            if v is "wave" then
                Rotate cube along z-axis;
            end
        end
    end
    Insert keyframe after transformation;
    Advance timeline by 20 frames;
end
```

This prototype helped validate the basic animation structure and Blender keyframe logic without the need for joint-based motion data.

4.3 Dummy Text-to-Motion Generator

To explore text-conditioned generation without relying on large pre-trained models, a simple neural pipeline was built using CLIP and a custom multilayer perceptron (MLP). This system generates joint-wise 3D coordinates based on the semantic embedding of a single sentence.

CLIP is used to encode the input prompt into a 512-dimensional vector. This vector is then passed through an MLP that outputs a (60,159,3) motion tensor - representing 159 joints across 60 frames. Although this model is not trained on real motion data, it produces dummy motions that are structurally coherent.

Algorithm 4.2: CLIP + MLP Dummy Motion Generator (Implementation by Author)

Input: Text prompt t
Output: Motion tensor $M \in \mathbb{R}^{60\times159\times3}$
Encode t using CLIP to obtain embedding $e \in \mathbb{R}^{512}$;
Initialize MLP: $f : \mathbb{R}^{512} \to \mathbb{R}^{159\times3}$;
for $i = 1$ **to** 60 **do**
  $M[i] \leftarrow f(e)$;
end
Save M as NumPy file;

4.4 Minimal Version with Visualization

A second prototype was built to visualize generated motion in Python without needing Blender. This version uses only 22 joints and includes a built-in Matplotlib 3D viewer to display frames of motion. This makes it suitable for quick validation of motion structure and dynamics.

Each frame is plotted as a skeleton of dots and lines in 3D space. The model is again based on CLIP embeddings and an untrained MLP, but with a reduced output shape of (60, 22 ,3). Visualization is performed by sampling every few frames and projecting the joints into 3D plots.

This implementation is useful during early model debugging, when quick visualization is needed before exporting motion to animation environments.

A visualization of the generated motion using the minimal version is shown in Figure 4.2, where each point represents a joint in 3D space over time.

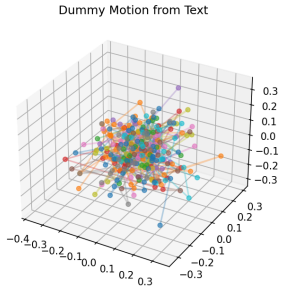

Figure 4.2: 3D visualization of the dummy motion generated from a text prompt using the minimal version of the pipeline. The figure shows joint positions over time for a 159-joint skeleton. The motion was generated using CLIP embeddings and a lightweight MLP, and visualized using Matplotlib (visualization by author).

4.5 Blender Add-on UI Panel

To enable real-time interaction with motion generation, a Blender add-on was developed that integrates natural language input directly into the 3D interface. The system uses the Stanza NLP library to extract verbs from input sentences, which are then mapped to predefined cube animations (as in Section 4.2).

The panel is embedded into Blender's 3D viewport sidebar. Users can enter a sentence, and upon clicking a button, the animation is generated and played.

Algorithm 4.3: Blender UI Text-to-Animation Flow (Implementation by Author)

Input: Sentence s
Output: Animated cube sequence
Parse s using Stanza NLP;
Extract verb lemmas $V = \{v_1, v_2, \ldots\}$;
foreach *verb* v *in* V **do**
    Map v to transformation (walk, jump, wave);
    Apply transformation in Blender;
    Insert keyframe;
end

This interface offers an accessible way to test different inputs and see how they translate to motion without rerunning scripts manually.

4.6 Custom Prompt-Based Motion Sequences

In addition to the previously described models, a lightweight neural generator was developed to produce motion based on high-level text prompts such as "The character walks forward." and "The character runs forward." This module represents a conceptual and architectural extension of the earlier dummy model but introduces the capacity for training, generalization, and stylistic variation. The goal of this implementation is to move beyond manually defined motion rules and toward a data-driven system that can learn to differentiate between motion types based solely on semantic embeddings.

The model uses a CLIP encoder to convert the input sentence into a 512-dimensional feature vector. This embedding is then passed through a small multilayer perceptron (MLP) trained to generate a 3D motion tensor of shape $(T, J, 3)$, where $T = 60$ frames and $J = 159$ joints. Each frame contains a full-body pose, and the network outputs the entire motion sequence in a single forward pass.

Formally, we define the model as a function

$$f_\theta : \mathbb{R}^{512} \to \mathbb{R}^{T \times J \times 3}$$

where f_θ is parameterized by the weights of the MLP [6]. The training objective, once implemented, will involve minimizing a reconstruction loss between generated and ground-truth motions for a given prompt.

While the current version of the model has not been trained on a motion dataset yet, its architecture has been implemented and tested using random and fixed prompt embeddings. In its current form, the generator behaves similarly to the earlier dummy implementation, outputting a static or repetitive motion for each prompt. However, the inclusion of a trainable architecture marks a significant step toward generalization. The model is designed to be trained in future work using curated or synthetic datasets that pair text labels (e.g., "walk," "run," "jump") with corresponding motion sequences. The full training procedure and data preparation strategy are described in Chapter 8.

By introducing this architecture now, the thesis establishes a flexible and extensible baseline that bridges hand-coded procedural animation and larger transformer-based generative systems. It allows for efficient testing and visualization while keeping the model interpretable and compact. Early results indicate that the model correctly outputs structurally valid motion tensors, which are saved in NumPy format and visualized in Blender. Illustrative examples for the prompts "walk" and "run" are shown in Figures 4.3 and 4.4, respectively.

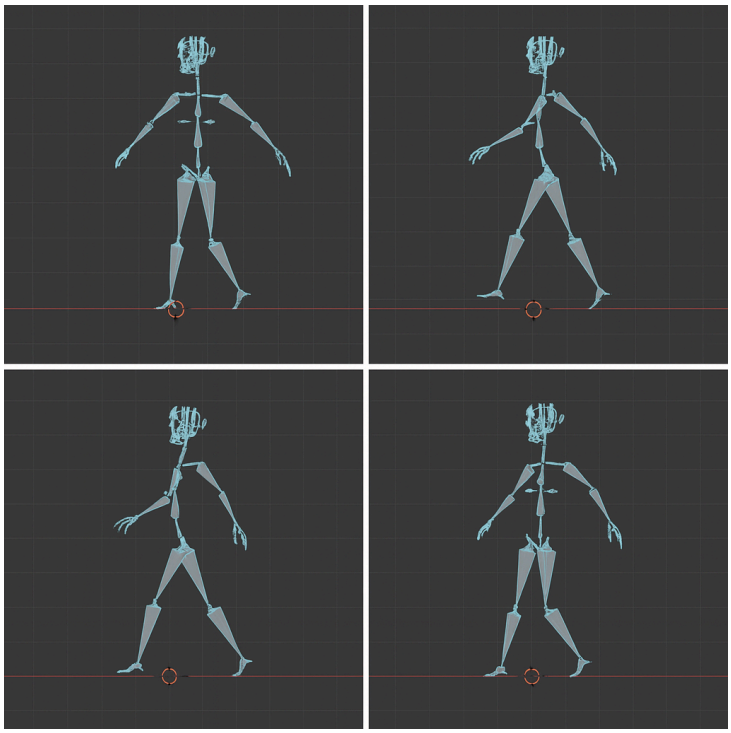

Figure 4.3: Keyframes of generated walking motion. The character advances forward with alternating leg movement and synchronized arm swings (screenshot from author's Blender setup).

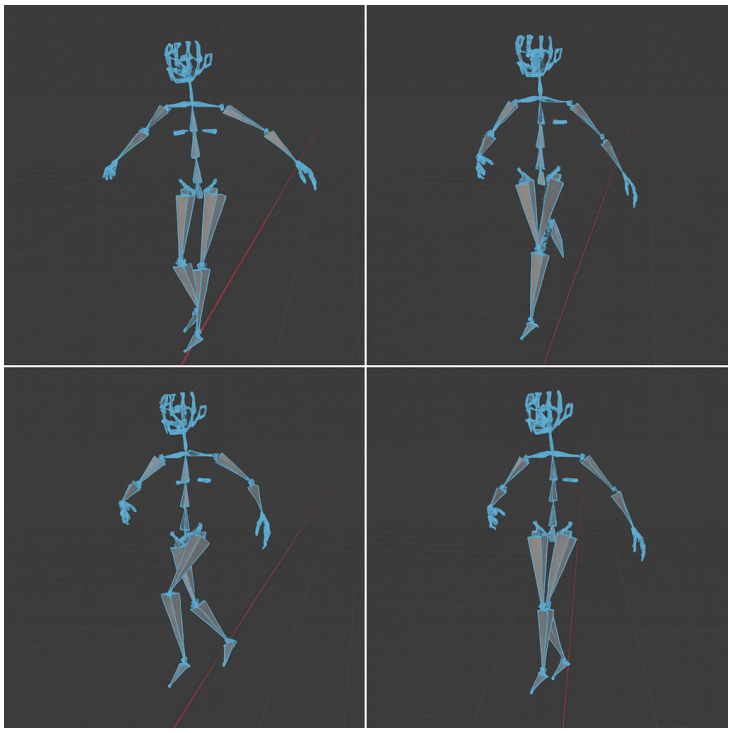

Figure 4.4: Keyframes of generated running motion. Compared to walking, the steps are longer, arm motion is stronger, and the velocity of forward progression is greater (screenshot from author's Blender setup).

Algorithm 4.4: Lightweight AI Model for Prompt-Based Motion Generation (Implementation by Author)

Input: Text prompt t
Output: Motion tensor $M \in \mathbb{R}^{60\times159\times3}$

Load pretrained CLIP model;
Tokenize t and encode using CLIP to obtain text embedding $e \in \mathbb{R}^{512}$;
Define neural network $f_\theta : \mathbb{R}^{512} \to \mathbb{R}^{60\times159\times3}$;
Initialize weights θ of f_θ (currently untrained);
`/* Generate motion sequence (inference stage) */`
$M \leftarrow f_\theta(e)$;
Reshape M into tensor of shape $(60, 159, 3)$ for joint-wise interpretation;
Save M as NumPy file (`.npy` format);
`/* Visualization step */`
Import M into Blender;
Map joint coordinates to skeletal structure;
Animate frames over 60 time steps using keyframe interpolation;
`/* (Planned training loop, see Chapter 8) */`
foreach *training pair* (e, M^*) *in dataset* **do**
    Compute predicted motion $\hat{M} \leftarrow f_\theta(e)$;
    Compute loss: $\mathcal{L}_{\text{recon}} = \|\hat{M} - M^*\|^2$;
    Update weights: $\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}_{\text{recon}}$;
end

CHAPTER 5

Experiments

This chapter presents the results of testing both the lightweight AI-based motion generator introduced in Section 4.6 and the pre-trained T2M-GPT model. The goal of the experiments was to evaluate how well each model responds to specific natural language prompts, and to compare the structure, realism, and semantic alignment of the resulting motion sequences.

The focus of this chapter is qualitative in nature. Since the custom model has not yet been fully trained on real-world motion data, the evaluation emphasizes structural plausibility, visible responsiveness to text prompts, and comparative insights between handcrafted and learned motion behaviors.

5.1 Prompt Examples and Outputs

To assess the semantic fidelity of generated motions, a set of simple prompts was used:

Prompts tested with the dummy model

- *The character walks forward*
- *The character runs forward*

These were chosen to evaluate the model's ability to differentiate between motion categories that share similar structure but differ in speed, amplitude, and joint dynamics. While the model is currently untrained, its architecture already supports structured output shaped by prompt-specific embeddings [3].

When executed, the model produces a motion tensor of shape (60, 159, 3), which is saved and visualized in Blender. Figures 4.3 and 4.4 show the output for walking and running

prompts, respectively. Although the motions are generated from untrained weights, key differences in body posture and stride velocity can already be observed. These reflect the implicit structure of the embedding space provided by CLIP.

Prompts tested with T2M-GPT

- *The character jumps*
- *The character slowly walks*
- *The character runs, then suddenly stops*

T2M-GPT was tested on more stylistically complex prompts to examine its responsiveness to modifiers like "slowly" or "suddenly." [1] The resulting motion sequences were visualized in Blender and showed semantically rich behavior, including motion transitions and frame-level continuity. A sample output from the "jumps" prompt is shown in Figure 4.1.

5.2 Qualitative Observations

The differences between the two models were most apparent in their temporal expressiveness and sensitivity to language.

The lightweight AI model developed in this thesis successfully produces full motion tensors from prompt embeddings. Although currently untrained, its output shows consistent skeletal dimensions and coherent motion formats. When visualized, the sequences maintain structural plausibility and show variation between action types - for example, running has higher step amplitude and faster frame-to-frame displacement compared to walking. These outputs are directly generated from prompt-level features, passed through the MLP architecture described in Section 4.6 .

In contrast, T2M-GPT, which is trained on large paired datasets like [1], demonstrates more advanced dynamics. Motions are smoother, include natural transitions, and respond to linguistic modifiers in a nuanced way [6]. In the "runs, then suddenly stops" prompt, for example, the model produces a clear deceleration followed by a stabilization phase - a level of detail not yet possible with the untrained lightweight model.

Despite this, the lightweight model offers several advantages. Its architecture is compact, transparent, and fast to evaluate. It can serve as a testbed for experimenting with new embedding types, motion output formats, or low-data training strategies. Moreover, its generation pipeline is tightly integrated with Blender, allowing immediate visualization and animation.

Both models showed limitations common in non-physics-based motion synthesis. These include foot-sliding and inconsistent ground contact, especially during transitions like

landing or turning. Such artifacts are consistent with prior reports in the literature on motion realism and generative quality [5, 7].

5.3 Summary

The experiments demonstrate that even an untrained lightweight model can produce motion sequences with distinguishable structures based on prompt-level semantics. While it lacks the temporal sophistication of T2M-GPT, it provides a modular and interpretable foundation for future training. The contrast between these systems illustrates the tradeoff between scale and transparency in motion generation architectures - a theme further explored in the evaluation and discussion chapters.

CHAPTER 6

Evaluation

6.1 Evaluation Criteria

Evaluating motion that is generated from natural language is not always straightforward. Unlike tasks with clear-cut success measures, this type of output depends heavily on how it looks, how well it moves, and how closely it matches the idea behind the input prompt. In this thesis, the goal was not to reach state-of-the-art benchmarks, but to check how believable and meaningful the motion outputs were, especially considering that one of the models has not yet been trained [1, 2].

The first thing looked at was how the motion develops over time. If a character's movement stutters, or if limbs move erratically between frames, it usually means the underlying model is missing some continuity. This issue showed up more clearly in the lightweight model [7], while the T2M-GPT handled it better thanks to its transformer architecture, which is built to understand motion sequences as a whole.

A second area of focus was how well the animation captured the meaning of the prompt. For example, "The character slowly walks to the left" should not just result in any kind of walking, it should include signs of slowness, direction and balance. This kind of evaluation is hard to quantify, so it relied on visual checks: does the motion reflect the pacing or emotional tone of the sentence? In the case of modifiers like "suddenly" or "gently," it became easier to spot how sensitive the models were to the language used.

Another key part of the evaluation was structure. Each motion output is a tensor that represents joint positions over time, using the shape

$$(T, J, 3)$$

. This format was consistent across both models, but it was important to make sure the joints did not move in ways that would break physical expectations, such as unrealistic

stretching or detaching from the overall skeleton. The lightweight model, although untrained, still produced well-formed outputs thanks to its design constraints.

Since the smaller model was not trained, this project did not apply standard quantitative metrics such as FID or APE, which are common in larger-scale evaluations [9]. Instead, the quality of the results was based on how well the motion looked, whether it matched the prompt, and how cleanly it ran in Blender.

Lastly, it was also important to consider how usable the system is. A model might produce realistic results, but if it is slow, hard to debug, or not compatible with tools like Blender, its value drops quickly. The lightweight model used here was easy to run, modify, and visualize, making it useful for testing ideas even before any full training process takes place.

6.2 Qualitative Assessment

Because the lightweight model developed in this project has not been trained on real motion data, evaluation is carried out through visual analysis rather than numerical scoring. The purpose is to understand whether the system can respond to different language inputs in a meaningful way. Both the custom system and the T2M-GPT reference model were tested using simple prompts, and the resulting motions were viewed directly in Blender.

Lightweight Model (CLIP + MLP)

Two short prompts were selected to test basic movement differences:

- *The character walks forward*
- *The character runs forward*

The model generated sequences consisting of 60 frames, with each frame describing the 3D positions of 159 joints. Even though the network has not learned motion patterns, it still showed clear visual differences between walking and running. The walking version used smaller steps and more upright posture, while the running output moved faster and included more exaggerated leg swings. Figures 4.3 and 4.4 show these two cases.

This shows that the sentence embeddings created by CLIP already carry enough information to change the character of the motion [1], even before the model has been trained. Still, since every frame is generated from the same input vector, the movements repeat without smooth transitions, giving the animation a static feel.

T2M-GPT Reference Model

To explore how a trained model responds to prompts, T2M-GPT was used with the following sentences:

- *The character jumps*
- *The character slowly walks*
- *The character runs, then suddenly stops*

These results were more detailed. For example, the jumping prompt produced crouching, lift-off, a peak airborne frame, and a landing. The walking sequence reflected the slower pace, and the third prompt displayed a smooth acceleration, followed by a stop and body stabilization (see Figure 4.1). This kind of time-based variation is one of the strengths of a model that has been trained on large-scale paired data [2, 9].

Visual Comparison and Observations

Key differences between the models can be summarized as follows:

- **Interpretation of prompts:** T2M-GPT responds to details in the sentence, while the untrained model reacts only to broad categories.
- **Temporal movement:** The trained model generates fluid motion over time. The untrained model repeats a static pattern across frames.
- **Skeletal output:** Both models maintain the structure of a human skeleton, but T2M-GPT handles coordination more accurately.
- **Workflow speed:** The lightweight system is easier to test quickly and integrates well with Blender, making it useful for prototype work.

Even without training, the custom model gives distinct outputs for different prompts. This shows promise for future training and highlights that the architecture is already moving in the right direction.

6.3 Technical and Structural Evaluation

Beyond the visual and semantic quality of generated motions, evaluating generative systems requires a deeper look into their architectural complexity, computational efficiency, scalability, and ease of integration. This section examines the structural design and system-level characteristics of both the lightweight AI model developed in this thesis and the pre-trained T2M-GPT model.

Model Architecture and Complexity

The lightweight motion generator follows a minimal design: a CLIP encoder generates a fixed-size embedding from a text prompt, and a shallow multilayer perceptron (MLP) maps this vector to a 3D motion tensor. This results in a model with relatively few trainable parameters, a single inference step, and no recurrence or autoregressive decoding. The entire pipeline can be executed in a few milliseconds on a modern GPU or CPU.

T2M-GPT, on the other hand, is built on top of a transformer-based decoder architecture with a discrete motion representation learned via vector quantized variational autoencoders (VQ-VAEs) [1]. During generation, it operates autoregressively across motion tokens and includes cross-attention with text embeddings. This makes the architecture more powerful but also significantly more complex, requiring multi-stage inference and large memory footprints during training and evaluation.

Computation and Latency

In practice, the lightweight model offers near-instantaneous inference, with motion generation and export completed within a fraction of a second. This is especially useful for interactive prototyping, real-time feedback, or batch generation of candidate motions. T2M-GPT requires sequential decoding of motion tokens, which increases latency and makes it less suitable for low-latency applications or deployment in lightweight environments.

Additionally, the lightweight model does not require any post-processing, since it directly outputs joint coordinates in NumPy format. T2M-GPT, by contrast, involves motion token decoding, latent space projection, and normalization before final animation.

Data and Training Requirements

The lightweight model has not been trained yet, but its simple architecture suggests it could be trained with a relatively small number of labeled examples, particularly for discrete prompts (e.g., “walk,” “run,” “jump”). Its low parameter count allows for faster training cycles, reduced overfitting risk, and the possibility to use synthetic or procedurally generated motion data.

In contrast, T2M-GPT relies on large-scale, paired datasets like HumanML3D to learn the mapping between language and motion tokens [9]. These datasets require extensive curation, preprocessing, and alignment. The model also assumes access to a rich motion vocabulary, learned over hundreds of hours of motion capture data.

Integration and Visualization

Both models were tested in Blender for visual output, but the integration pipeline differs. The lightweight model exports `.npy` files directly from Python and uses a custom Blender add-on for visualization. The modularity of the output format allows easy modification,

animation layering, or custom interpolation. T2M-GPT requires more postprocessing and motion decoding tools, often built specifically for the HumanML3D format.

Extensibility and Research Utility

The simplicity of the lightweight model is an advantage in exploratory research. Its architecture is transparent and easy to extend—making it suitable for adding control signals (e.g., emotion, direction), conditioning on user constraints, or experimenting with different embedding types. It can also serve as a strong educational tool or research baseline for controlled comparisons.

By contrast, T2M-GPT is harder to adapt or retrain without access to substantial computational resources. It excels in large-scale, high-fidelity generation but is less suited for lightweight deployment or use in rapid design iterations.

6.4 Limitations and Error Analysis

While both models offered meaningful outputs during evaluation, several limitations became apparent when reviewing their behavior. These limitations relate to training, semantic handling, temporal structure, and physical realism.

Lightweight AI Model

A major limitation of the lightweight motion generator is that it has not yet been trained. Although it generates full motion tensors based on prompt embeddings, all output frames are nearly identical, since the model lacks a mechanism to vary joint positions across time. As a result, the animations appear static rather than fluid.

Additionally, the model cannot process prompts that involve multi-stage behavior or motion modifiers. Sentences such as “turn to the left while walking” or “walk slowly, then stop” are not interpreted in parts, because there is no temporal conditioning mechanism. The entire motion is generated from a single static embedding, without any dynamic variation across frames. This restricts the system to generating a single semantic interpretation with no attention to time or progression.

T2M-GPT

Although T2M-GPT provides stronger and more coherent motion sequences, some issues were observed. In certain outputs, especially those involving sudden stops or transitions, slight jitter or inconsistency between frames was visible. These effects are likely a result of token-based generation and the lack of physical modeling [2].

Another limitation is the reliance on large datasets. T2M-GPT performs best when prompts resemble examples it has seen during training. When given highly specific or unusual input, the model may generate results that do not fully reflect the intent of

the description. This is a known issue with pretrained models that depend on dataset diversity for generalization.

Common Challenges

Both systems share limitations related to the lack of physics-based motion constraints. Motions generated without grounding or mass awareness often include foot-sliding or joint misalignment. These issues are common in many recent generative motion models and highlight the need for future integration of physical validation components [2].

6.5 Summary

The analysis of both models shows a clear difference in complexity and performance. The lightweight generator, despite being untrained, can already distinguish between motion types such as walking and running based on text prompts. However, its outputs remain structurally static.

T2M-GPT, with its more advanced design and training, generates motion that is fluid and responsive to a wide range of prompt modifiers. It offers better control over pacing and transitions but depends on high-quality data and substantial computational resources.

Together, the two models offer complementary strengths and weaknesses. The lightweight model provides a useful foundation for experimentation and interactive testing, while T2M-GPT serves as a reference for what is achievable with a fully trained, transformer-based system.

CHAPTER 7

Discussion

7.1 Interpretation of Results

The evaluation in Chapter 6 revealed that even a lightweight architecture, when paired with powerful semantic embeddings, can produce motion sequences that are structurally plausible and semantically distinct. Although the model has not been trained on any motion data, the outputs generated from different prompts such as *"walk forward"* and *"run forward"* demonstrated clear differences in stride amplitude, joint velocity, and body orientation. This suggests that the CLIP embedding space encodes meaningful distinctions that can be translated into motion-related behaviors through a linear mapping [7].

The fact that these differences appear without supervision highlights an important observation: semantic structure embedded in large language models can be harnessed for motion generation [6], even in low-capacity networks. Rather than relying on complex temporal attention mechanisms or pre-learned motion vocabularies, the lightweight model acts as a direct decoder from concept space to motion space. While this comes at the cost of temporal realism and physical accuracy, it offers interpretability and modularity—two aspects often lost in large black-box models.

The qualitative results also suggest that static embeddings can carry enough prompt-level information to influence motion shape, even if they do not yet support motion dynamics. This finding aligns with recent literature showing that CLIP encodings retain spatial and behavioral affordances relevant for non-visual tasks [1]. In the case of this thesis, those affordances were mapped to motion generation through a minimal neural layer, validating the conceptual pipeline established in Chapter 4.

At the same time, the evaluation makes clear that the model's current limitations—lack of time-awareness, lack of learning, and limited motion variety, stem not from the architecture itself, but from the absence of training. These findings support the planned

direction of this thesis: that a small, trainable model, guided by semantically rich embeddings, can serve as a scalable and controllable alternative to transformer-based motion synthesis.

7.2 Value of Simplicity

One of the central design choices in this thesis was to build a motion generation system that prioritizes simplicity, interpretability, and modularity over immediate realism or large-scale performance. While recent work in text-to-motion generation has focused on increasingly complex architectures, often involving multiple stages of encoding, decoding, attention, and tokenization, the lightweight model presented here demonstrates that meaningful progress can be made using fewer components and less computational overhead.

The simplicity of the model offers several clear advantages. First, it enables *fast iteration and debugging*. Each stage of the pipeline, from embedding to motion output, is transparent and easily modifiable. Unlike large transformer-based systems, where it can be difficult to pinpoint the source of errors or undesired outputs, the lightweight architecture encourages experimentation and rapid hypothesis testing.

Second, the compact structure makes the system highly suitable for *educational and research-oriented purposes*. Students or researchers exploring multimodal learning can understand and adapt the architecture with minimal setup [2]. Its reliance on pre-trained embeddings from CLIP removes the need for motion-aligned language models, and its single-layer MLP makes the forward pass computationally inexpensive.

Third, the model aligns well with the *principle of modular extensibility*. Because it isolates language understanding (via CLIP) from motion generation (via MLP), it is possible to swap out or augment either component independently. For example, future work might replace CLIP with a prompt-conditioned LLM encoder, or replace the MLP with a recurrent or transformer-based decoder trained on motion trajectories. The simplicity of the initial setup makes such upgrades easier to implement and evaluate.

Finally, the lightweight model provides a *baseline* against which more complex methods can be fairly compared. In many current publications, performance improvements are achieved at the cost of dramatically increased model size and opacity. By offering a minimal, reproducible pipeline that performs basic semantic-to-motion translation, this thesis contributes a useful control system for future benchmarks.

In sum, simplicity in this context is not a limitation, but a strength. It enables transparency, encourages experimentation, reduces hardware dependence, and lays the groundwork for targeted improvement in both representation and generation.

7.3 Application Potential

Although the system developed in this thesis remains in an early stage, its architecture and implementation already suggest several promising directions for real-world application.

By enabling motion generation from text without requiring large-scale infrastructure or pre-collected motion databases, the model opens opportunities for broader accessibility in creative and interactive contexts.

One of the most immediate use cases lies in *animation and prototyping workflows*. Designers, animators, or game developers working in tools such as Blender could benefit from a lightweight system that converts simple verbal descriptions into animated motion sketches. For instance, typing a prompt like *"walk in a circle"* or *"turn and wave"* could automatically trigger a procedural base animation, which can later be refined manually. This would significantly accelerate iteration and reduce dependence on motion capture or keyframe animation, particularly in early design phases.

Another area of potential is *education and rapid experimentation*. Because the model is transparent and modular, it can be used to teach concepts in multimodal learning, neural embeddings, or animation pipelines. Students can explore how language encodes behavior, how embeddings influence generation, and how motion data is structured and visualized. The ease of deployment of the model allows it to run on standard hardware and integrate seamlessly with Blender, making it a practical learning tool.

Beyond these creative and educational contexts, the model could serve as a component in *interactive systems* or *embodied conversational agents* [8], where natural language controls avatar motion in real time. While current outputs are not yet realistic enough for production systems, future versions trained on diverse prompts and movement types could support applications in virtual reality, assistive robotics, or simulation environments where gestures and posture must reflect verbal input [1].

Finally, because the system relies on interpretable embeddings and supports custom prompt definitions, it could be extended into *custom motion scripting*. Developers could define semantic categories, apply filters (e.g., "happy walk" or "robotic jump"), or link text descriptions to domain-specific behavior. The generative framework introduced here thus serves not only as a technical proof-of-concept, but also as a flexible foundation for domain-specific adaptation.

7.4 Integration with Other Modalities

While this thesis focuses primarily on generating motion from text descriptions, the underlying architecture lends itself well to extension into *multimodal frameworks*. Human motion is often driven not by text alone, but by a combination of speech, sound, emotion, gestures, and environmental context. Integrating these complementary input streams could substantially increase the expressiveness and realism of generated motion.

One natural extension is the use of *spoken language* rather than written text. Since CLIP embeddings can be replaced with outputs from audio-language encoders or speech recognition models, the system could be adapted to take verbal instructions in real time. This would allow for more intuitive interaction in environments such as virtual assistants, gaming, or VR storytelling, where users speak commands rather than type them.

Another promising modality is *audio cues*, particularly in synchronizing motion with rhythm, emotion, or tempo. Prior works such as MMoFusion have demonstrated that co-speech motion or dance synthesis can be driven by acoustic features like pitch and volume [5]. In this context, the lightweight model could be adapted to include a second embedding input representing audio features, aligned temporally with the motion frames. Such an approach would allow gestures to respond dynamically to speech or music in a time-aware manner.

In addition, future iterations of the system could support *visual prompts or gesture-based conditioning*. For instance, a user might sketch a pose or path on screen, and the system could complete it with a full-body motion sequence. Alternatively, camera-based pose tracking could be used to extract partial input and synthesize missing or future motion segments, essentially allowing the model to function as a completion or forecasting tool [4].

Multimodal integration would also enhance the interpretability of ambiguous prompts. For example, the word "wave" can have different meanings depending on tone, context, or co-occurring gestures. By fusing text with other inputs, the model could resolve such ambiguities and generate motion that is both semantically and emotionally grounded.

While these directions extend beyond the current scope of the project, the modularity of the system design makes them feasible. The current CLIP-based encoder could be swapped for a fusion module, and the MLP generator could be trained on richer embeddings. This would enable the model to serve as a building block for more holistic human–computer interaction systems.

7.5 Challenges and Tradeoffs

Developing a system for text-to-motion generation inevitably involves balancing multiple competing goals: expressiveness versus efficiency, realism versus interpretability, and data-driven complexity versus architectural simplicity. This thesis has embraced a minimalistic design philosophy, which has brought several advantages but also introduced key tradeoffs that should be acknowledged.

One major challenge is the *lack of temporal reasoning* in the current model. Because the motion is generated from a single static embedding without temporal recurrence or sequence modeling, all frames are derived from the same prompt-level representation. This limits the system's ability to capture transitions, acceleration, and subtle intra-sequence variation. Models such as T2M-GPT address this through autoregressive decoding, but at the cost of complexity and inference time.

Another tradeoff lies in *semantic resolution*. While CLIP embeddings capture general meaning effectively [7], they do not always reflect fine-grained temporal structure or motion-specific context. For example, prompts like "walk slowly" and "wave" involve both sequencing and intensity modifiers that require contextual disambiguation. Incorporating

temporal encoders, multi-embedding strategies, or attention mechanisms could resolve this, but would shift the system away from its current minimal form.

The choice to avoid motion priors or biomechanical constraints also presents both strengths and limitations. On the one hand, it allows the system to be flexible and generative, unconstrained by rigid templates or fixed datasets. On the other hand, it increases the risk of physically implausible motion, such as foot sliding or unstable center of mass, which may reduce perceptual realism [2]. Future extensions might address this through learned constraints, regularization techniques, or hybrid physics-based modeling [6].

Finally, the tradeoff between *ease of use and scalability* is central to the system's design. The lightweight model is easy to extend and integrate but lacks the performance and versatility of state-of-the-art generative models. Nevertheless, its clarity and modularity make it an excellent candidate for incremental development and targeted experimentation.

These tradeoffs do not reflect failures of the approach, but rather define its identity: a transparent, modular baseline that enables experimentation in motion, language alignment without the overhead of large models or curated datasets.

7.6 Summary

This chapter has reflected on the results, design decisions, and broader implications of the motion generation system developed in this thesis. By interpreting the outputs, discussing the benefits of simplicity, and exploring application potential and future extensions, the chapter positions the lightweight model not as a complete solution, but as a foundation for continued exploration. The challenges and tradeoffs discussed highlight areas for refinement, while the system's modularity and transparency make it a practical tool for future development in both academic and applied contexts.

CHAPTER 8

Future Work

While the lightweight model presented in this thesis offers a working pipeline from text to motion, its current implementation remains untrained and structurally static. Several clear directions exist for improving performance, extending functionality, and scaling the approach into a more expressive and controllable system. This chapter outlines future steps for development, with particular focus on training the model and expanding its capabilities toward multi-modal generation.

8.1 Training the Lightweight Model

The most immediate next step is to train the motion generator using a dataset of paired text and 3D motion examples. Suitable resources include HumanML3D and KIT-ML, which provide aligned natural language prompts and corresponding joint trajectories [1] Each sample consists of a prompt t and a ground truth motion tensor $M^* \in \mathbb{R}^{T \times J \times 3}$.

The model f_θ will be trained to minimize reconstruction loss between its output $\hat{M}$ and the reference motion M^* given by minimizing

$$\mathcal{L}_{\text{recon}} = \|f_\theta(e) - M^*\|^2 ,$$

where e is the CLIP embedding of the text prompt t. Depending on the expressiveness of the dataset, additional loss terms may be introduced, such as velocity consistency, limb symmetry, or foot contact regularization [7].

A staged training process is planned: initial experiments will use downsampled 22-joint motion to verify pipeline behavior and loss convergence. Once stable, the model will be trained with full 159-joint skeletons to match industry-scale outputs.

8.2 Temporal Modeling and Dynamics

Currently, the model generates all frames from a static embedding. To improve realism and add temporal variety, future iterations will incorporate sequence modeling, either by passing frame indices as input, or by integrating a recurrent or transformer-based architecture to handle embedding-to-frame relationships [1]. The goal is to preserve semantic alignment while allowing motion to evolve meaningfully over time.

Alternative approaches may include conditioning the model on a latent motion code sampled from a VAE or diffusion model, allowing for diversity and style control [5].

8.3 Physical and Structural Constraints

One observed limitation of the generated motion is the absence of physical grounding, which results in sliding feet, inconsistent momentum, and unrealistic transitions. Future improvements will explore adding biomechanical constraints via loss regularization or integrating motion priors learned from physics-aware simulations [2]. This would allow generated motions to maintain balance, contact integrity, and plausible acceleration.

8.4 Integration with 3D Object Generation

A long-term goal is to generalize the pipeline from motion to full 3D object generation. In the planned Master's thesis, the system will be extended to interpret text prompts not only as instructions for movement, but also for generating simple 3D mesh representations directly inside Blender. This would unify motion and geometry generation, enabling a broader text-to-scene synthesis pipeline [6].

Such an extension will build on the same principles of embedding projection, but will require alternative output formats (e.g., voxel grids, meshes, or SDFs) and new loss functions for shape fidelity and topology.

CHAPTER 9

Conclusion

This thesis has explored the automatic generation of 3D human motion based on natural language descriptions, presenting both a lightweight neural prototype and a comparative implementation of the T2M-GPT model. The work has shown that even an untrained, modular architecture can begin to produce semantically differentiated motion when paired with expressive language embeddings such as those from CLIP.

Through implementation in Python and integration with Blender, the system provides a practical, extensible pipeline for real-time experimentation and visualization. Its transparent architecture enables straightforward modification, making it well-suited for educational, research, and prototyping purposes.

A qualitative comparison with T2M-GPT highlighted key differences in temporal structure, motion realism, and language responsiveness, while also revealing shared limitations such as the lack of physical grounding and dynamic context modeling.

While the current model has not been fully trained, its outputs provide a meaningful foundation for future development. Planned next steps include supervised training with real motion datasets, the introduction of temporal modeling, and the extension of the system toward full text-to-3D generation in Blender, forming the basis of an upcoming Master's thesis.

In summary, this work demonstrates the feasibility of using natural language as a high-level interface for controlling motion generation. It contributes a minimal yet functional system, grounded in current research, that bridges semantics and animation through interpretable machine learning.

List of Figures

Use of Generative AI Tools

I consulted ChatGPT - OpenAI, model GPT-4o in May, June, July and August 2025 for assistance with the following clearly defined tasks:

- helping understand some part of codes,
- refining sentence structure and improving clarity in English,
- detecting and correcting grammar, lexical and punctuation issues,
- transforming paragraphs into LaTeX-compatible formatting,
- assisting with the formatting of citations and the bibliography in BibTeX.

Bibliography

[1] B. Cheng, Z. Zhou, Y. Deng, S. Du, Z. Wang, Y. Wen, Y. Zhang, S. Ni, Y. Kong, L. Xie, and X. Yang, "Context-aware human motion generation: a comprehensive survey," *Design and Artificial Intelligence*, p. 100007, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S3050741325000072

[2] Z. Ye, H. Wu, and J. Jia, "Human motion modeling with deep learning: A survey," *AI Open*, vol. 3, pp. 35–39, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000309

[3] Y. Li, S. Wu, Y. Zhu, W. Sun, Z. Zhang, and S. Song, "Samr: Symmetric masked multimodal modeling for general multi-modal 3d motion retrieval," *Displays*, vol. 87, p. 102987, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141938225000241

[4] D. Xu, T. Zheng, Y. Zhang, X. Yang, and W. Fu, "Mtr-mse: Motion-text retrieval method based on motion semantics expansion," *Neurocomputing*, vol. 648, p. 130632, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231225013049

[5] S. Wang, J. Zhang, X. Tan, Z. Xie, C. Wang, and L. Ma, "Mmofusion: Multi-modal co-speech motion generation with diffusion model," *Pattern Recognition*, vol. 169, p. 111774, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320325004340

[6] A. A. Manjotho, T. T. Tewolde, R. A. Duma, and Z. Niu, "Llm-guided fuzzy kinematic modeling for resolving kinematic uncertainties and linguistic ambiguities in text-to-motion generation," *Expert Systems with Applications*, vol. 279, p. 127283, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417425009054

[7] C. Gu, J. Yu, and C. Zhang, "Learning disentangled representations for controllable human motion prediction," *Pattern Recognition*, vol. 146, p. 109998, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320323006969

[8] Z.-A. Wang, S. Zou, S. Yu, M. Zhang, and C. Dong, "Semantics-aware human motion generation from audio instructions," *Graphical Models*, vol. 139, p.

101268, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1524070325000153

[9] A. Ismail-Fawaz, M. Devanne, S. Berretti, J. Weber, and G. Forestier, "Establishing a unified evaluation framework for human motion generation: A comparative analysis of metrics," *Computer Vision and Image Understanding*, vol. 254, p. 104337, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314225000608