Lifelike Motions for Robotic Characters

AGON SERIFI

2025

DISS. ETH NO. 31327

DISS. ETH NO. 31327

Lifelike Motions for Robotic Characters

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES
(Dr. sc. ETH Zurich)

presented by
AGON SERIFI
born on 12.05.1997

accepted on the recommendation of
Prof. Dr. Markus Gross, examiner
Dr. Moritz Bächer, co-examiner
Prof. Dr. Michiel van de Panne, co-examiner

2025

Abstract

Humanoids have made significant advances in recent years. Nonetheless, the motions they perform often remain rigid, mechanical, and lack the diversity and expressiveness of human motion. This stands in stark contrast to physics-based simulated characters, which are capable of performing agile and lifelike motions in fully simulated environments. Such characters typically leverage reinforcement learning in combination with motion capture data to learn how to move like humans. However, their success is closely tied to unrealistic modeling assumptions such as simplified dynamics, overpowered actuators, or noise-free sensing. While these assumptions enable efficient and stable training, they hinder the transfer to the real world.

In the real world, there are no shortcuts. To achieve more dynamic motions for humanoids, physically accurate simulation and robust learning methods are essential. This requires rethinking many components along the pipeline, starting from the simulators and how to account for sim-to-real gaps, up to questions about how to represent, track, and generate motions for humanoids.

In this dissertation, we present several contributions in this direction and bring more lifelike motions to robotic characters. First, we present a learning-based modular simulation augmentation to reduce the sim-to-real gap. Our method can generalize across robot configurations and helps to better estimate the state of the robot. In a second contribution, we propose a novel architecture for encoding motions as a trajectory in latent space. The architecture overcomes the need for absolute positional encoding, leading to better reconstruction quality of various sequential data types. In a third contribution, we show how a pretrained latent space can be leveraged to train more accurate and robust control policies using reinforcement learning. Our two-stage method transfers to the real world and brings dynamic dancing motions to a humanoid robot. Our last contribution physically aligns kinematic motion generators with the capabilities of the character and its control policy. This allows for a more successful transfer of generated motions to the real world.

The methods and concepts introduced in this dissertation make robots move more lifelike and reduce the gap to simulated characters. We hope they will inspire future research and bring more believable robots into our world.

Zusammenfassung

Humanoide Roboter haben in den letzten Jahren bemerkenswerte Fortschritte gemacht. Dennoch bleiben die von ihnen ausgeführten Bewegungen häufig starr, mechanisch, und ihnen fehlt die Vielfalt und Ausdruckskraft menschlicher Bewegungen. Dies steht in starkem Kontrast zu physikbasierten simulierten Charakteren, die in der Lage sind, agile und lebensechte Bewegungen in vollständig simulierten Umgebungen auszuführen. Solche Charaktere nutzen typischerweise Reinforcement Learning in Kombination mit Motion-Capture-Daten, um zu lernen, sich wie Menschen zu bewegen. Ihr Erfolg ist jedoch eng an unrealistische Modellierungsannahmen geknüpft, wie vereinfachte physikalische Modelle, übermässig starke Aktuatoren oder ideale Sensorik. Zwar führen diese Annahmen zu effizientem und stabilem Training, erschweren jedoch die Übertragung in die reale Welt.

In der realen Welt gibt es keine einfachen Abkürzungen. Um dynamischere Bewegungen von Humanoiden zu erreichen, sind präzise physikalische Simulationen und robuste Lernverfahren unerlässlich. Dies erfordert ein Umdenken in vielen Teilen der gesamten Pipeline—von den Simulatoren und der Berücksichtigung der Sim-to-Real-Lücke bis hin zu grundlegenden Fragen der Repräsentation, des Trackings und der Generierung von Bewegungen für humanoide Roboter.

In dieser Dissertation stellen wir mehrere Beiträge vor, die darauf abzielen, die Bewegungen robotischer Charaktere lebensechter zu gestalten. Zunächst präsentieren wir eine lernbasierte, modulare Erweiterung der Simulation, die die Lücke zwischen Simulation und Realität verringern soll. Unsere Methode ist auf verschiedene Roboterkonfigurationen anwendbar und erhöht die Genauigkeit der Zustandsschätzung. Im zweiten Beitrag präsentieren wir eine neuartige Architektur, die Bewegungen als Trajektorien in einem latenten Repräsentationsraum modelliert. Diese Architektur kommt ohne absolute Positionskodierung aus und erzielt eine höhere Rekonstruktionsqualität für unterschiedliche sequenzielle Datentypen. Im dritten Beitrag zeigen wir, wie ein vortrainierter latenter Repräsentationsraum verwendet werden kann, um durch Reinforcement Learning genauere und robustere Policies zu erlernen. Die zweistufige Methode lässt sich in die reale Welt übertragen und ermöglicht einem humanoiden Roboter dynamische Tanzbewegungen. Unser letzter Beitrag passt kinematische Bewegungsgeneratoren physikalisch an die Fähigkeiten des Charakters und seine Poli-

cy an. Dadurch können die generierten Bewegungen in die reale Welt übertragen werden.

Die in dieser Dissertation vorgestellten Methoden und Konzepte ermöglichen Robotern lebensechtere Bewegungen und verringern die Lücke zu simulierten Charakteren. Wir hoffen, dass sie zukünftige Forschung inspirieren und dazu beitragen, überzeugendere Roboter in unsere Welt zu bringen.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to Dr. Moritz Bächer for his mentorship. Thank you for guiding and supporting me, for your trust, and for providing feedback on all early-stage ideas. Your tireless dedication and passion, experience, and knowledge have played an important role in not only shaping this work but also me as a researcher. You have set an example as a leader and visionary, I strive to embody in my career.

I want to extend my gratitude to Prof. Dr. Markus Gross, whose remarkable work has created a world-class research environment. I am thankful to have been part of the Computer Graphics Lab and the opportunity to complete my Bachelor's, Master's, and Ph.D. in your group. Thank you for sharing your academic wisdom during these past nine years at ETH Zürich. What you have accomplished—and continue to accomplish—inspires me deeply.

I am grateful to Dr. Espen Knoop for all the robots he designed and built, and all the times he repaired them. Thank you for your guidance and fruitful discussions throughout, and for showing me how to densify my work, although I decided to ignore that learning in this section. I admire your efficiency and precision in everything you do.

Furthermore, I wish to thank Dr. Ruben Grandia for his clear thoughts and structured discussions. Thank you for asking the critical questions early on and for showing me how to approach hard problems. I admire the breadth of knowledge you have across so many fields.

It is my honor to thank Prof. Dr. Michiel van de Panne for being on my committee. This thesis builds on many inspiring contributions from your group.

During my studies, I had the chance to work with excellent supervisors who encouraged me in pursuing a research career. I learned a lot from all of them. Thank you, Prof. Dr. Nikolina Ban and Prof. Dr. Tobias Günther, for a great Bachelor's thesis, which led to my first publication. I want to thank Dr. Prashanth Chandran, Dr. Gaspard Zoss, Dr. Paulo Gotardo, and Dr. Derek Bradley for supervising my Master's thesis; your knowledge and guidance, along with the experience I gained during this time, were instrumental and set an inspiring example of what it means to be a great researcher.

Thank you, Lima, for enduring the early failures, getting back up every time, and nailing the final shot just in time for the deadlines. I also want to thank the anonymous reviewers for their invaluable feedback, which helped improve our work. And to the reviewer who asked whether Lima can do a backflip—*Can you?*

I want to thank many more *humans* who have been part of my journey, some of whom have made direct contributions to the work presented as co-authors or in discussions.

The remaining members of the Robotics Group at Disney Research—Christian Schumacher, David Müller, Georg Wiedebach, Guirec Maloisel, Josefine Klintberg, Maike Paetzel-Prüsmann, Michael Baumgartner, Michael A. Hopkins, Naveen Kumar, Sammy Christen, Vassilios Tsounis—for the great time. It has been amazing to see the team grow in size and in the unique contributions each person brings.

I am grateful for the friendships that have formed along the way. Thank you, Prashanth Chandran, for late-night brainstorming sessions. Kārlis Briedis, for stressing together about SIGGRAPH. Yingyan Xu, for always staying motivated—everything will be fine. Manuel Kansy for random kitchen discussions. Guirec Maloisel for forcing me to take the stairs. Gaspard Zoss for sharing his wisdom. David Müller for keeping me hydrated with coffee. Sammy Christen for showing me how to submit to arXiv. Roberto Azevedo for trying to convince me to go home earlier. Thank you, Pascal Chang, Rajesh Sharma, Sergio Sancho, and Xianyao Zhang, for the time together as teaching assistants, and for keeping the place alive after 6 pm. Thank you, Markus Portmann and Martin Nett, for taking care of the organizational tasks. Andreas Baumann for equipping me with the latest GPUs. Thank you to the remaining members of Disney Research for giving me a glimpse into the House of the Mouse. Thank you to all members of the Computer Graphics Lab, for Pizza-All-Hands, a nice retreat and bowling night.

Finally, I am immensely grateful for the unconditional support of my family. Your values and lessons shaped who I am today. My father, Zilkefi, introduced me to computer science and taught me how to solve and reason about problems. Thank you for sparking my curiosity in mathematics, physics, and programming. My mother, Sebahat, taught me the value of perseverance and dedication. Thank you for shaping my sense of patience and consistency. I am thankful for my amazing brother, Gent, who was by my side throughout, sharing the same passion and having a relentless work ethic. Your brilliance keeps me inspired and resilient. Thank you for consistently encouraging me to stay motivated and determined. I would not be where I am without the strength my family brings to every moment—*thank you*.

Preface

Artificial intelligence has redefined many fields. Yet, the moment embodied intelligence takes steps in our world, it looks and feels anything but alive.

Although decades of progress in *robotics* have led to strong and precise robots, current humanoids still do not move nearly as effortlessly as human beings.

Meanwhile, *computer graphics* has perfected the art of motion for virtual characters in fully simulated worlds.

—Therefore, the imminent question is how to teach intelligent robotic characters to move in ways that feel lifelike.

Contents

Abstract iii

Zusammenfassung v

Acknowledgements vii

Preface ix

Contents xi

List of Figures xv

List of Algorithms xvii

List of Tables xviii

Introduction 1
- 1.1 Topics and Contributions . . . 7
  - 1.1.1 Simulation . . . 8
  - 1.1.2 Representation . . . 8
  - 1.1.3 Tracking . . . 8
  - 1.1.4 Generation . . . 9
- 1.2 Publications . . . 10

Neural Simulation Augmentation 11
- 2.1 Related Work . . . 13
- 2.2 Overview . . . 14
- 2.3 Modular Augmentation of Simulation Representations . . . 15
  - 2.3.1 Decomposing Dynamics Simulations of Robots . . . 15
  - 2.3.2 Measurements . . . 17
  - 2.3.3 Reference Motions . . . 17
  - 2.3.4 Problem Statement . . . 17
- 2.4 Transformer-based Augmentation . . . 18
  - 2.4.1 State Augmentation . . . 19
  - 2.4.2 Training . . . 20

2.5 Evaluation and Results . . . 20
2.5.1 Actuator Modeling . . . 21
2.5.2 Rigid Body Modeling . . . 23
2.5.3 Augmentation Limits . . . 24
2.5.4 Ablation Studies . . . 25
2.6 Concluding Remarks . . . 30

Sequential Data Representation **33**
3.1 Related Work . . . 35
3.2 Spline-based Transformers . . . 37
3.2.1 Background: Splines . . . 37
3.2.2 Network Architecture . . . 38
3.3 Evaluation and Results . . . 40
3.3.1 Synthetic Datasets . . . 41
3.3.2 Images . . . 42
3.3.3 Animation . . . 43
3.3.4 Geometric Representation . . . 47
3.3.5 Practical Limitations . . . 48
3.3.6 Latent Space Visualization . . . 48
3.4 Concluding Remarks . . . 49

Robust Motion Tracking **51**
4.1 Related Work . . . 53
4.2 Two-Stage Processing . . . 55
4.2.1 Extracting Latent Motion Priors . . . 56
4.2.2 Training Conditional Policy . . . 58
4.3 Evaluation and Results . . . 59
4.3.1 Characters, Dataset, and Training Procedure . . . 59
4.3.2 Evaluation of Motion Embedding . . . 64
4.3.3 Evaluation of Two-Stage Processing . . . 65
4.3.4 Directability . . . 69
4.3.5 Robot Control . . . 70
4.4 Concluding Remarks . . . 71

Robot Motion Generation **73**
5.1 Related Work . . . 75
5.2 Method . . . 77
5.2.1 Critic Training . . . 78
5.2.2 Physics-Aligned Generative Model . . . 80
5.3 Evaluation and Results . . . 81
5.3.1 Kinematic Motion Generation . . . 82
5.3.2 Physical Alignment . . . 86

5.3.3 Physics-Based Motion Tracking 86
5.4 Concluding Remarks 89

Conclusion **91**
6.1 Summary and Limitations 91
6.2 Future Research 93

Additional Details: Spline-based Transformers **97**
A.1 More About Control Points 97
A.2 Implementation Details 98
A.3 Additional Results 101

References **105**

Declaration of the use of AI-based tools **123**

List of Figures

1.1 Thesis Overview. 7

2.1 Neural Simulation Augmentation Overview. 11
2.2 Modular Simulation Augmentation. 16
2.3 State Augmentation Transformer (SAT). 18
2.4 KickBot and DanceBot. 21
2.5 Actuator Augmentation Evaluation - Position. 23
2.6 Actuator Augmentation Evaluation - Electric Current. 24
2.7 Actuator Augmentation Trajectories. 24
2.8 Rigid Body Augmentation Evaluation. 25
2.9 Augmentation Limits. 26
2.10 Augmentation Robustness. 27
2.11 Attention over 128 history states. 29
2.12 SAT Validation Loss. 29

3.1 Spline-based Transformers Overview. 34
3.2 Variations of Latent Spaces. 40
3.3 Visual Synthetic Curves Reconstruction. 42
3.4 Visual Image Reconstruction. 43
3.5 Visual Facial Animation Reconstruction. 45
3.6 Full Body Reconstruction and Modification. 46
3.7 Spline-based Transformers - Training Performance (AFHQ). 48
3.8 Latent Splines (AFHQ). 49

4.1 Robust Motion Tracking. 51
4.2 Robust Motion Tracking Overview. 55
4.3 Latent Motion Similarity. 63
4.4 Tracking Performance. 66
4.5 Tracking Robustness. 68
4.6 Spatial Composition. 69
4.7 Motion editing. 70
4.8 Lima Robot. 71

5.1 Robot Motion Diffusion Model. 73
5.2 Robot Motion Diffusion Model Overview. 77

5.3 Motion Realism Comparison. 84
5.4 Realistic Motion Generation. 85
5.5 Collision Avoidance. 87
5.6 Robot Control on Generated Motions. 88
5.7 Robot Precision on Generated Motions. 89

A.1 Latent Space Similarities. 98
A.2 Modifying Control Points. 99
A.3 Additional reconstruction results for 2D curves. 102
A.4 Additional reconstruction results for test images. 103
A.5 Additional reconstruction results for test hairstyles. 104

List of Algorithms

1 Critic Training. 80

2 Spline-based Transformer. 99

3 Evaluation of Cubic Bézier. 100

List of Tables

2.1 Architecture Evaluation on KickBotA. 27

3.1 Synthetic Curves Reconstruction. 41
3.2 Image Reconstruction. 43
3.3 Face Performance Reconstruction. 44
3.4 Human Motion Reconstruction. 45
3.5 Strand Reconstruction. 47

4.1 Actuator, VAE, and RL Parameters. 61
4.2 Latent Space Comparison. 65
4.3 Ablation Study - Pose. 65
4.4 Scaling of VAE and RL component. 67

5.1 Training Parameters for Critic Training and Physical Alignment. . . 83
5.2 Kinematic Motion Generation. 83
5.3 Generated Motion Tracking. 88

A.1 Parameters and Hyperparameters. 101

CHAPTER 1

Introduction

Humans have long been fascinated by mechanical embodiments of the human form, which has driven efforts in building humanoids for centuries. A *humanoid* is a robot designed to resemble a human being in various characteristics, such as having an anthropomorphic body or performing human-like motions. Today, humanoid robotics has become a rapidly evolving field, as many of the required technologies and building blocks have matured. On the hardware side, we have seen improvements in components such as electric actuators, high-frequency IMUs, high-capacity batteries, or powerful onboard computers, enabling us to build lightweight, high-torque humanoids. Hand in hand, advances in building fast and accurate simulators, together with a paradigm shift towards learning-based methods such as reinforcement learning, have made it more efficient and cost-effective to experiment and acquire new behaviors. These new capabilities have led to a surge of innovation, with both academic and industrial institutions actively developing new humanoids at an astonishing pace.

When it comes to motion capabilities, research in the field has primarily focused on functional motion skills, such as locomotion. While doing so, the main focus has been on achieving robustness across diverse terrains, a goal that has many challenges of its own. The walking style and agility have been secondary concerns. Consequently, the motions humanoids perform often remain stiff, mechanical, and unnatural. They often rely on predefined gaits, repetitive patterns, and over-simplified motions with limited coordination, quickly revealing a clear gap from human-like behavior. Although there are no natural references of how a robot should move, we can intuitively distinguish between lifelike and artificial motions. This intuition has

been successfully implemented in computer animation, where virtual characters appear natural and expressive, thereby overcoming the uncanny valley of motion.

While the methods for controlling, generating, and imitating kinematic motions achieve remarkable human-like characters in simulated environments, we face additional challenges in extending these capabilities to the real world. Robots have physical constraints and limitations and must operate under the unforgiving laws of physics. While this may seem obvious, its implications are often underestimated; simply changing the simulation model is insufficient. What works in simulation might not translate to the real world. Learning algorithms can become slow in converging or collapse entirely. Moreover, motions that appear visually appealing might have dynamic issues and be physically infeasible.

In the context of this thesis, we aim to address the gap between physically embodied robots and physically simulated characters. The methods developed support both ends, and we refer to them in combination as *robotic characters*. We begin by building a narrative around the current state of the art in robotic character motion, revisiting the components that have led us to this point. Followed by a more in-depth discussion of the investigated topics and corresponding contributions of this thesis that lead to more dynamic and lifelike motions for robotic characters.

Humanoids Building humanoids has a long history. Starting with simple mechanical automata, such as Leonardo da Vinci's design for a mechanical knight in the late 15th century, humans built increasingly more complex systems that mimic human behavior. Pierre Jaquet-Droz, for example, designed automated dolls that could play instruments or write text with a pen. These early humanoids were driven by mechanical innovation and reflect a persistent desire to replicate not just the form but also the motion of human beings. Today, with improvements in hardware components and compute density, modern robots have become highly complex systems equipped with powerful computing units.

Model Predictive Control. Humanoids have to coordinate multiple body parts simultaneously to walk, jump, or even dance. Achieving this behavior requires sophisticated control frameworks that have to deal with whole-body dynamics, including momentum, ground contacts, friction forces, and other complex aspects of physical interaction. Traditionally, Model Predictive Control (MPC [Garcia et al., 1989]) has been used to solve the *continuous*

control problem. At every control step, one has to decide which *real-valued* control inputs to apply to the robot. Hereby, we aim to minimize a cost function, *e.g.*, staying close to a target position, while considering the current system dynamics and any constraints, such as joint limits. However, to do so reliably, MPC depends on an accurate model of the dynamics and is sensitive to any errors or inaccuracies in the system. When performing highly dynamic motions with unknown foot contact patterns, slippage, sensor noise, or rapidly changing states, the model's inaccuracy increases, resulting in failures and falls.

Reinforcement Learning. In the meantime, Reinforcement Learning (RL [Sutton and Barto, 2018]) has gained significant momentum in other fields, ranging from gaming [Schrittwieser et al., 2020] to the alignment of Large Language Models [Ouyang et al., 2022]. It has surpassed human-level performance in many well-studied tasks and found new solutions or strategies, for example, in chess. At its core, RL deals with the question of how an *agent* can learn to maximize a *reward* by interacting with an *environment*. At each time step, the agent observes the (partial) state of the environment, *e.g.*, the configuration of the chess pieces. The environment defines the rules, dynamics, and limitations under which the agent is operating. It defines the states an agent can be in, the actions that can be performed, and the responses that an action causes. The agents then learn to solve tasks solely through *experience*; what actions in which states were more successful than others. However, the current RL learning algorithms are sample-inefficient and require many trials and errors, which is why they first evolved in such discrete space, game-like environments that can play billions of games simultaneously. Therefore, to consider RL for robotic control, it is essential to perform a large number of trials. Since this is not feasible in the real world due to costs and scalability issues, one critical ingredient is access to fast, accurate, and highly parallelizable physics simulators.

Simulation. The history of physics simulators is crucial for the current progress in robotics and RL. Early general-purpose rigid body simulators evolved around the 2000s. Their underlying physics engine, such as the Open Dynamics Engine [Smith, 2005], employed hard contact models, where contacts were treated as instantaneous impulses, resulting in discontinuities and instabilities during simulation. Later, by sacrificing perfect rigidity and adopting smooth and soft contact models, as done in MuJoCo [Todorov et al., 2012], it became possible to achieve much greater simulation stability and accurate modeling of complex multi-body systems. To-

gether with highly optimized sparse solvers, MuJoCo became a fast and realistic simulator, enabling real-time simulation of robots. However, its CPU-based implementation set limits to scaling, as it could only simulate a couple of tens of environments simultaneously. With the recent rise of fully GPU-accelerated physics engines, such as PhysX, simulators like Isaac Gym [Makoviychuk et al., 2021] and its successor, Isaac Sim, have significantly increased simulation throughput, unlocking new possibilities through large-scale RL training pipelines.

RL for Continuous Control. Hand in hand with this improvement in simulation, major breakthroughs in continuous control of physics-based characters with RL were achieved. At the forefront, OpenAI Gym [Brockman et al., 2016] and DeepMind Control Suite [Tunyasuvunakool et al., 2020], with support for MuJoCo, started to standardize RL environments. With the introduction of benchmarks, RL algorithms began to evolve. An important step was the transition to Actor-Critic methods, which combine the impractical Value-Based methods (*e.g.*, Q-Learning [Watkins and Dayan, 1992]) with the inefficient policy gradient methods (*e.g.*, REINFORCE [Williams, 1992]), resulting in more stable and sample-efficient training. Among the Actor-Critic methods, the introduction of Proximal Policy Optimization (PPO [Schulman et al., 2017]) marked a major turning point, offering a practical and robust learning algorithm suitable for many different continuous control environments. Even today, despite the rapid progress in the field, PPO remains the go-to algorithm.

Physics-based Character Control. With improved learning algorithms, more complex tasks, such as locomotion for *physics-based humanoid characters*, could be addressed. This refers to characters with human proportions and movement produced by applying torques or forces at the joints within a physics simulator. Initially, the goal was to teach a *controller* to walk in a commanded direction at a specified velocity. Some natural walking behavior automatically emerges from such a task, but has many artifacts, such as waving arms and uncontrolled hops [Heess et al., 2017]. DeepMimic [Peng et al., 2018a] took one step further and combined the task learning with an imitation objective. Given a reference motion from a human motion capture setup, the character is tasked to fulfill a goal, such as reaching a point, while also following the reference motion as closely as possible. The resulting controller is capable of generating smooth, natural-looking behavior from just these simple objectives. This idea of learning from human motion capture data represents a significant step toward achieving lifelike general behav-

ior. However, most work focuses on purely simulated physics-based characters with spherical artificial joints. These are joints that can apply high torque values that are not feasible with current actuators. Sometimes, fictitious forces are applied to overcome the gap between underactuated or simplified characters and the ground-truth human body. Furthermore, the characters have simplified collision volumes, making the simulation more efficient. While those characters can already perform impressive motions, the techniques presented are not directly applicable or guaranteed to work in the real world. Furthermore, the techniques suffer from scalability issues and require heavy resources even in their simplified setup. This is why robots are still unable to match the performance of simulated characters.

Learning from Motion Data. The increased availability of large motion capture datasets such as AMASS [Mahmood et al., 2019], enabled more and more Deep Learning (DL [LeCun et al., 2015]) applications. One central goal in DL is to learn compact, informative representations of the data by extracting relevant features and mapping them into a latent space. Leading methods for this include Variational Autoencoders (VAEs [Kingma and Welling, 2013]), which can be built on various architectural backbones, most commonly convolutional layers [LeCun et al., 1989] or transformer-based models [Vaswani et al., 2017]. These backbones are not explicitly designed with motion data in mind, but they have been shown to be effective on this data type as well [Holden et al., 2015]. Given such a latent space that is both compact and semantically meaningful, we can draw new latent codes and decode them into new data samples. This simple procedure marks the beginning of generative AI.

Generative AI. In the last few years, Generative AI has taken over the world. Today, we can generate text, code, images, and, more recently, video. Among the first models to generate high-fidelity data samples are Generative Adversarial Networks (GANs [Goodfellow et al., 2014]). Those models are trained using a generator and a discriminator network that play a game against each other. The discriminator tries to distinguish between generated samples and real ones, treating any input that does not belong to the dataset as fake, while the generator tries to produce data that fools the discriminator. While impressive results can be achieved, those models are difficult to train and unstable, often producing only simple, low-diversity samples—a common issue known as mode collapse. More recently, Diffusion Models [Ho et al., 2020] have demonstrated remarkable versatility across a range of data types. These models estimate the gradient of the log-probability of the data

distribution, also known as the score function, to gradually shift a noise distribution towards the true data distribution. This process is closely related to Langevin dynamics [Langevin, 1908], a method from statistical physics for sampling complex distributions. More recently, these models were also adapted to the motion space and demonstrated remarkable text-to-motion capabilities [Tevet et al., 2023]. However, while the generated motions might look visually appealing, they often do not consider physicality, such as dynamic balancing, forces acting on the body, or joint limits. Given these limitations, they have not been widely deployed in robotics so far.

Challenges. Bringing a wide range of human motions to robots, to make them move more dynamically and be more versatile, is an open problem with many challenges. The physical world is unforgiving, requiring control strategies that remain stable in the presence of uncertainty and system variability. Actuation is imperfect, measurements are noisy, and latency in control loops is critical. The motion data, captured from humans, is not tailored for robots, which lack the flexibility and dynamic range of a human body. All these factors impose constraints that prevent the use of unrealistic assumptions during training, increase simulation complexity, reduce learning efficiency, and ultimately require us to develop new methods to overcome them.

The combination of hardware, reinforcement learning, GPU-accelerated physics simulators, motion capture data, and advances in motion representation has suddenly made this challenge plausible—and the following pages present steps in the direction of bringing more lifelike robotic characters into our world.

1.1 Topics and Contributions

This thesis investigates and presents contributions in four components for achieving lifelike behavior in robotic characters. Starting from accurate physical modeling to sophisticated generative models:

1. *Simulation:* Accurate simulation models are crucial for reliably predicting the robot's states to ensure the observations align with the real world.
2. *Representation:* Motion data representation is important to fully leverage deep learning capabilities.
3. *Tracking:* Robust and general motion tracking abilities built the essential foundation for lifelike motions.
4. *Generation:* Generative methods provide ways to automate and create motions for robots in more natural ways through high-level commands.

The dissertation presents each topic and contribution in its own chapter, each starting with a focused introduction and discussion of related work, and ending with concluding remarks. Finally, in Chapter 6 we summarize our work, present conclusions, and discuss existing limitations. Furthermore, we explore possibilities for future work and motivate directions for subsequent research.

In the following, we introduce and highlight the contributions in every topic and Chapter. Fig. 1.1 provides an illustration of the different topics and how they connect with each other. Note, however, that while we assigned each chapter to a primary component, the methods in a single chapter often address multiple components simultaneously. Further, we do not present a fully closed integrated system, but improvements for each component.

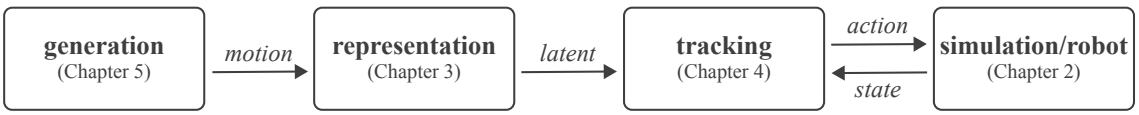


Figure 1.1: ***Thesis Overview.*** *Illustration of the topics and their connection. In Chapter 5 we present a motion generator. Chapter 3 introduces a novel architecture to map motions into latent representations. Chapter 4 contributes a tracker that leverages pretrained latent spaces to track motion references accurately. And in Chapter 2 we show how neural networks can improve the simulation accuracy of robot states.*

1.1.1 Simulation

Simulation representations of robots have advanced in recent years. Yet, significant sim-to-real gaps remain due to modeling assumptions and hard-to-model effects such as friction.

In Chapter 2, we propose to augment common simulation representations with a transformer-inspired architecture, by training a network to predict the actual state of a robot building block given its simulated state. Because we augment building blocks, rather than the full simulation state, our approach is modular and improves generalization and robustness. We use our method to augment the mechanical and electrical state of robot actuators, as well as the state of rigid bodies.

Our actuator augmentation generalizes well across robots, and our rigid body augmentation results in improvements even under high uncertainty in model parameters.

1.1.2 Representation

Deep neural networks are used to extract features from high-dimensional data spaces into lower-dimensional latent spaces. The current state-of-the-art architectures are transformer-based; their underlying attention mechanism can attend between any two data points in time. However, during this process, the data sequence is transformed into a single latent token, losing any clear notion of position.

In Chapter 3 we introduce Spline-based Transformers, a novel class of Transformer models. Inspired by workflows using splines in computer animation, our Spline-based Transformers embed an input sequence of elements as a smooth trajectory in latent space.

We demonstrate the superior performance of Spline-based Transformers in representing sequential data on a variety of datasets, ranging from synthetic 2D to large-scale real-world datasets of images, 3D shapes, and animations.

1.1.3 Tracking

Recent progress in physics-based character control has made it possible to learn policies from human motion capture datasets. However, it remains challenging to train a single control policy that works with diverse and unseen motions and can be deployed to real-world physical robots.

In Chapter 4, we propose a two-stage technique that enables the control of a character with a full-body kinematic motion reference, with a focus on imitation accuracy. In a first stage, we extract a latent space encoding by training a variational autoencoder, taking short windows of motion from unstructured data as input. We then use the embedding from the time-varying latent code to train a conditional policy in a second stage, providing a mapping from kinematic input to dynamics-aware output. By keeping the two stages separate, we benefit from self-supervised methods to get better latent codes and explicit imitation rewards to avoid mode collapse.

We demonstrate the efficiency and robustness of our method in simulation, with unseen user-specified motions, and on a bipedal robot, where we bring dynamic motions to the real world.

1.1.4 Generation

Recent advancements in generative motion models have achieved remarkable results, enabling the synthesis of lifelike human motions from textual descriptions. These kinematic approaches, while visually appealing, often produce motions that fail to adhere to physical constraints. We, therefore, observe a *gen-to-real gap* leading to motions that a tracking controller can not perform, resulting in artifacts that impede real-world deployment.

To address this issue, we introduce in Chapter 5 a novel method that integrates kinematic generative models with physics-based character control. Our approach begins by training a reward surrogate to predict the performance of the downstream control task, offering an efficient and differentiable loss function. This loss is then employed to fine-tune a baseline generative model, ensuring that the generated motions are not only diverse but also physically plausible for real-world scenarios. The outcome of our processing is the Robot Motion Diffusion Model (RobotMDM), a text-conditioned motion diffusion model that interfaces with a reinforcement learning-based tracking controller.

We demonstrate the effectiveness of this method on a challenging humanoid robot, confirming its practical utility and robustness in dynamic environments.

1.2 Publications

In the context of this thesis, the following peer-reviewed work has been published:

> **Agon Serifi**, Espen Knoop, Christian Schumacher, Naveen Kumar, Markus Gross, and Moritz Bächer. Transformer-based neural augmentation of robot simulation representations. *IEEE Robotics and Automation Letters*, 8(6):3748-3755, 2023.
>
> Prashanth Chandran*, **Agon Serifi***, Markus Gross, and Moritz Bächer. Spline-based transformers. In *Computer Vision – ECCV 2024 (Oral)*, pages 1–17, 2024.
>
> **Agon Serifi**, Ruben Grandia, Espen Knoop, Markus Gross, and Moritz Bächer. Vmp: Versatile motion priors for robustly tracking motion on physical characters. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '24, pages 1–11, 2024.
>
> **Agon Serifi**, Ruben Grandia, Espen Knoop, Markus Gross, and Moritz Bächer. Robot motion diffusion model: Motion generation for robotic characters. In *ACM SIGGRAPH Asia 2024 Conference Papers*, SA '24, pages 1–9, 2024.

During the course of this thesis, the following peer-reviewed work was also published:

> Lucas N. Alegre*, **Agon Serifi***, Ruben Grandia, David Müller, Espen Knoop, and Moritz Bächer. Amor: Adaptive character control through multi-objective reinforcement learning. In *ACM SIGGRAPH 2025 Conference Papers*, SIGGRAPH '25, 2025.
>
> Sammy Christen, David Müller, **Agon Serifi**, Ruben Grandia, Georg Wiedebach, Michael A. Hopkins, Espen Knoop and Moritz Bächer. Autonomous Human-Robot Interaction via Operator Imitation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

* denotes equal contribution.

CHAPTER 2

Neural Simulation Augmentation

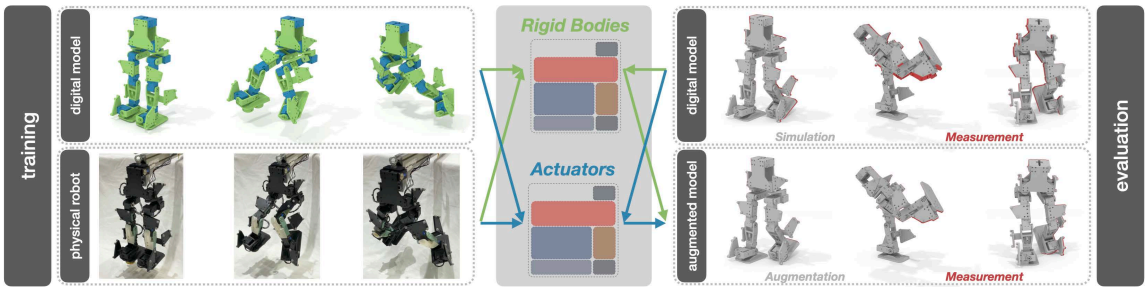


Figure 2.1: ***Overview.*** *As input, our processing takes reference motion that is sent to a physical robot, and also a digital model thereof (left). The physical behavior of the robot results in partial measurements of its time-varying state, while simulations (digital model) provide full state estimates. We first sort the different components, with their corresponding simulated and measured state, into categories (rigid bodies in green, actuators in blue). We then train a State Augmentation Transformer (SAT) per category, minimizing a loss that penalizes differences between augmented and measured states (middle). To apply augmentations to simulations of robots (right), we proceed analogously (reference motion, digital model), unifying the individual augmentations after evaluation (augmented model).*

This chapter is based on the publication *"Transformer-Based Neural Augmentation of Robot Simulation Representations"*, Agon Serifi, Espen Knoop, Christian Schumacher, Naveen Kumar, Markus Gross, and Moritz Bächer [Serifi et al., 2023]. A supplemental video is available at `https://youtu.be/DMAg-I6F5XA`.

In robotics, simulation plays a role of ever-increasing importance. Yet, even with state-of-the-art techniques, discrepancies between simulations and reality (sim-to-real gaps) are observed. This is in part due to modeling assumptions (*e.g.*, no deformation of rigid bodies, no backlash, no friction), and in part due to model inaccuracies (*e.g.*, errors in rigid body mass properties, tolerances during assembly). Actuator drives may often also implement control loops where the detailed implementation is undisclosed, making them difficult to model accurately. While additional verification and characterization experiments can reduce the sim-to-real gap, this adds complexity and does not scale well. In this chapter, we instead tackle the problem with a learning-based data-driven approach.

Concretely, we propose a transformer-based architecture that augments simulation representations of individual building blocks robots are made of. For each class of building blocks, we train a separate State Augmentation Transformer (SAT), taking the time-varying state and interaction forces with other entities as input. We minimize the sim-to-real gap with a physics-informed loss that compares augmented simulation states to measurements taken from physical robots. As we demonstrate with several examples, the augmented state consistently improves the prediction quality, both when augmenting *actuator* states and *rigid body* states.

Our actuator augmentation generalizes well across robots that are built with the same actuators. Training a single augmentation for all rigid components of a robot, we show that our augmentation improves simulation prediction even under high uncertainty in model parameters. Note that due to the higher dimensionality of rigid bodies, the rigid body augmentation does not currently generalize across different robots.

Succinctly, our technical contributions are:

- a transformer-based neural augmentation of simulation representations for a large class of robots consisting of rigid components, mechanical joints, and actuators.
- a modular augmentation approach that interfaces with all common simulation representations of dynamical systems and could be extended to other robot building blocks.

Our method enables accurate digital twin representations of robots, with applications including more accurate state estimation, and improved closed- and open-loop control. While we here augment rigid components and actuators, we keep our formal description general and would expect our method and modular approach to also extend to other building blocks.

2.1 Related Work

Our transformer-based augmentation shares similarities with architectures commonly used in natural language processing and time series forecasting [Torres et al., 2021; Lim and Zohren, 2021]. We first review related architectures, followed by a discussion of neural simulation representations.

Neural Architectures for Time Series Forecasting. Recurrent Neural Networks (RNNs) [Rumelhart et al., 1986] have been applied to time series forecasting tasks, with Long Short-Term Memory (LSTM) cells seeing widespread use [Hochreiter and Schmidhuber, 1997; Bianchi et al., 2017]. However, their limited short-term memory [Bengio et al., 1994; Pascanu et al., 2013] is insufficient for simulation augmentation as we demonstrate with a series of experiments.

Transformers [Vaswani et al., 2017] overcome these limitations with an attention mechanism, providing long-term memory. While transformers lead to state-of-the-art performance on problems ranging from natural language [Vaswani et al., 2017] to image [Dosovitskiy et al., 2021] and audio [Baevski et al., 2020] processing, we have not seen the use of this architecture in augmentations of simulation representations. We base our architecture on the Temporal Fusion Transformer (TFT) proposed by Lim *et al*. [2021]. We confirm that a combination of short- and long-term memory outperforms an attention-only architecture, and is well-suited for the problem domain we consider here.

Neural Simulation. Differentiable simulation [Giftthaler et al., 2017; Carpentier and Mansard, 2018; de Avila Belbute-Peres et al., 2018; Todorov et al., 2012; Hu et al., 2020; Geilinger et al., 2020] enables a tight integration of simulation and data-driven techniques. For example, Geilinger *et al*. [2020] integrate a differentiable simulator as a last node of a neural network to train control policies. However, differentiable simulators are not widely available. An advantage of our technique is that it interfaces with a wide range of standard simulators, augmenting states of building blocks robots are made of.

Golemo *et al*. [2018] train an RNN to predict differences between simulated and real-world behavior and integrate it with the simulator to learn more robust patterns. Although differentiable simulators allow gradient-based methods for parameter fitting, they are limited by the underlying approximate model. Hence, it is impossible to fully close the sim-to-real gap. Recently, data-driven methods were introduced to learn additional nonlinear

dynamics, where neural networks are used to enhance the simulators. Kloss *et al*. [2020] integrate a neural network to augment the input of an analytic model into the simulation pipeline. This approach, however, keeps the simulation within the space of the analytical model and is therefore still restricted. On the other hand, Ajay *et al*. [2018] apply a variational RNN [Chung et al., 2015] as a post-process, allowing the augmentation to go beyond the expressiveness of an analytic model. Heiden *et al*. [2021] present a hybrid simulator combining a physics engine with a neural network to augment simulation variables. Their approach is trained in an end-to-end fashion, specializing in specific tasks. In contrast to previous works, we focus on a decoupled augmentation of simulation states and show that this modular approach can generalize over different robots assembled from the same building blocks. Additionally, our transformer-based model augments the simulation states while consuming a more extended history of previous states, mitigating error accumulations over time.

2.2 Overview

We consider here the dynamic simulation of robots made of rigid components which are driven by actuators and may also be coupled together with mechanical joints.

Training. During the training phase (Fig. 2.1, left), we send representative reference motions to the physical robot and its digital model, resulting in partial measurements and simulations of the time-varying state of the robot.

We then group the building blocks into categories. For most robots, we define two categories: one for rigid components (Fig. 2.1, in green), and one for actuators (in blue). We then train a separate SAT (middle) for each category, by minimizing a loss that penalizes differences between augmented simulation states and corresponding measurements.

Evaluation. To augment simulations of the same or a different physical instance of the same robot performing a different task or motion (right), we proceed analogously: We first simulate the robot's time-varying behavior, then group the robot's components into the same categories as defined during training (digital model). We then evaluate the SATs for each component separately, resulting in an augmentation of the full state of the robot at every time step (augmented model).

2.3 Modular Augmentation of Simulation Representations

Before delving into the specifics of our transformer architecture, we will discuss how we prepare simulation data and measurements for training.

To allow for generalization, we propose to decompose a simulation representation into building blocks, then learn an augmentation for *all* building blocks of a particular type or class. To decompose simulation representations, we utilize that the robot is in an equilibrium at every time step. This method scales well because we can augment *any* robot that is made of the same building blocks. Moreover, the technique is extensible because we can easily add new SATs for new types of building blocks.

2.3.1 Decomposing Dynamics Simulations of Robots

A dynamic simulation computes the time-varying state of the robot, along with the forces and torques on each component, such that it is in equilibrium at any point in time. The interactions between components manifest themselves as forces and torques at the actuators and joints. To isolate a component, we, therefore, consider its state along with the computed time-varying forces and torques acting on it (in green in Fig. 2.2).

We here interface with a maximal-coordinate simulation representation [Erez et al., 2015], to allow for robots with arbitrary kinematics, including series-parallel structures. However, our approach could also be readily used with a reduced-coordinate simulation formulation.

The mechanical behavior of an individual building block can be represented with position and velocity quantities, $\mathbf{p}(t)$ and $\mathbf{v}(t)$, uniquely describing its state $\mathbf{s}(t)$. For rigid bodies, the position quantities are the pose of the body, and the velocity quantities consist of its linear and angular velocities. For actuators, we are interested in the position and velocity of the output shaft, whose state is fully described with a 2D vector.

Building blocks are coupled with a set of constraints $\boldsymbol{\mathcal{C}}$, implementing the degrees of freedom of mechanical joints and actuators. For rigid components, the constraint forces $\mathbf{f}_{\boldsymbol{\mathcal{C}}}(t) = \boldsymbol{\mathcal{C}}_{\mathbf{p}}^{T}\boldsymbol{\lambda}$, with Lagrange multipliers $\boldsymbol{\lambda}(t)$, enable the decoupled simulation of the building block.

Succinctly, in this chapter, we seek to learn an augmentation $\Delta\mathbf{s}(t)$ of the time-varying states $\mathbf{s}(t)$ to account for modeling uncertainties in the equa-

tions of motion

$$\begin{array}{rcl} \dot{\mathbf{p}} & = & \mathbf{v} \\ \dot{\mathbf{v}} & = & \mathbf{M}^{-1}(\mathbf{p})(\mathbf{f}(\mathbf{p},\mathbf{v}) + \mathbf{f}_{\mathcal{C}}) \end{array} \text{ with state } \mathbf{s} = \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \end{bmatrix}, \tag{2.1}$$

of mechanical components, with generalized mass matrix $\mathbf{M}$, forces $\mathbf{f}$ that model gravity, damping and other body forces, and constraint forces $\mathbf{f}_{\mathcal{C}}$ that we extract from a simulation.

Rather than training an augmentation for each building block in isolation, we seek to train a network that outputs an augmentation $\Delta\mathbf{s}(t)$ when fed with $\mathbf{s}(t)$ and $\mathbf{f}_{\mathcal{C}}(t)$ of building blocks of a particular class. For example, we seek to train a single SAT that generalizes across *all* rigid bodies. This is possible as long as all building blocks in a particular class have the same number of state variables and constraint force components (*e.g.*, six for rigid bodies). Note that if there are robots that consist of components of vastly different sizes, it could make sense to train several SATs for a particular class.

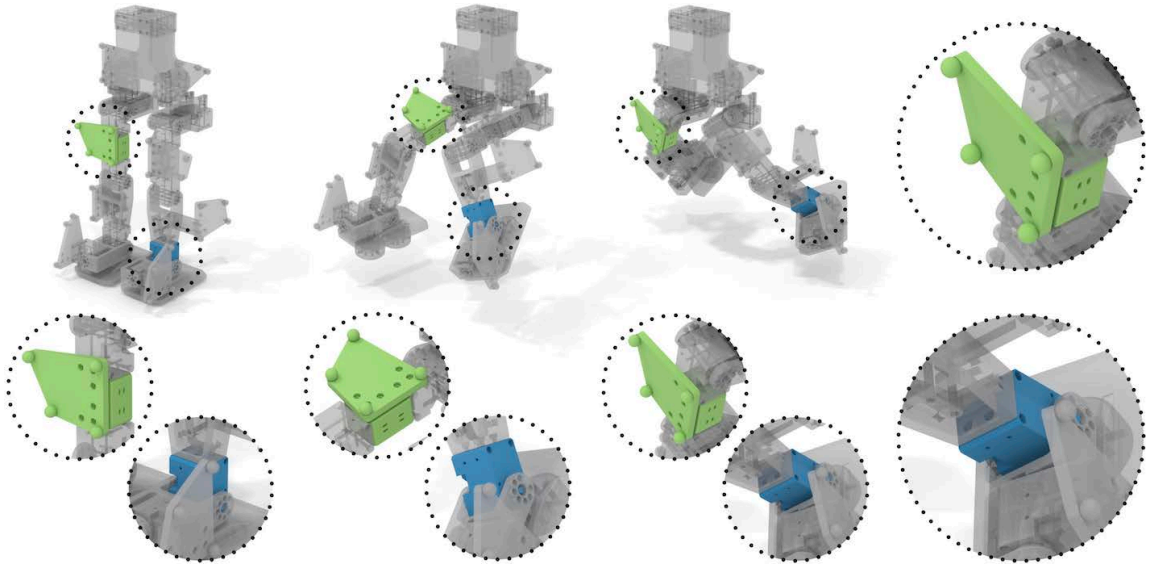


Figure 2.2: ***Modular Augmentation.*** *We decouple the motion of building blocks a robot is made of by recording the time-varying forces and torques that neighboring building blocks exert on them. By doing so, we could simulate their motion in isolation.*

In our simulation, we model actuators with a PID controller and a standard brushed DC motor model. The state of an actuator is described by its (1D) position, velocity, current, and voltage, and it takes as input reference values $\mathbf{r}(t)$ of position, velocity, and torque. Analogously to rigid bodies, we augment classes of actuators that we can represent with the same number of state variables. To this end, we record the constraint forces, which in this

context represent the dynamics of the part of the robot the actuator is driving. In contrast to a mechanical system, however, our neural augmentation takes the reference curves $\mathbf{r}(t)$ as additional inputs.

2.3.2 Measurements

Our actuators use built-in sensors to measure their position, velocity, voltage, and current, which we use for training our actuator SATs. To train augmentations of the mechanical components, we add motion markers (four spheres on the green component in Fig. 2.2, zoom-in, top, right), then track them using a commercial tracking system to obtain the time-varying rigid body trajectory. These data sources result in partial measurements $\bar{\mathbf{s}}(t)$ of the full state.

2.3.3 Reference Motions

To collect a representative and sufficiently large dataset, we draw end-effector trajectories from randomly generated B-Spline curves of varying order $(0-4)$. We then use an inverse kinematics formulation [Schumacher et al., 2021] to generate the reference curves $\mathbf{r}(t)$ that we send to the simulator and the physical robot.

2.3.4 Problem Statement

For every class of building blocks, our neural augmentation takes decoupled states $\mathbf{s}(t)$, constraint forces $\mathbf{f}_{\mathcal{C}}(t)$, and partial state measurements $\bar{\mathbf{s}}(t)$ as input, learning an augmentation $\Delta\mathbf{s}(t)$ that minimizes the difference between the augmented $\mathbf{s}(t) + \Delta\mathbf{s}(t)$ and measured state $\bar{\mathbf{s}}(t)$.

To measure differences between states and partial measurements, we introduce a constant matrix $\mathbf{S}$ that selects and transforms full-state quantities so that we can numerically compare them to measurements. In loss functions, we then compare $\mathbf{S}\left(\mathbf{s}(t) + \Delta\mathbf{s}(t)\right)$ to $\bar{\mathbf{s}}$. To avoid unnecessary transformations during training and evaluation, we preprocess the data, subtracting the transformed state, $\mathbf{S}\mathbf{s}(t)$, from the measurements, then comparing

$$\Delta\bar{\mathbf{s}}(t) = \bar{\mathbf{s}} - \mathbf{S}\mathbf{s}(t) \quad \text{to} \quad \mathbf{S}\Delta\mathbf{s}(t) \tag{2.2}$$

in loss functions.

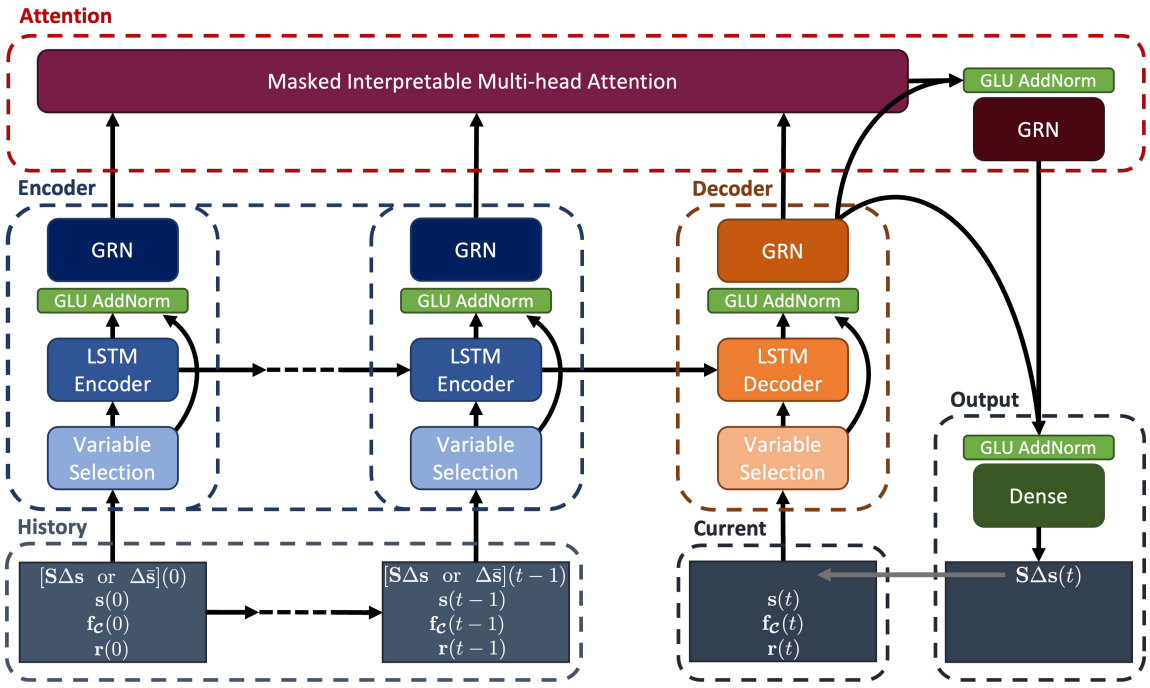


Figure 2.3: ***State Augmentation Transformer (SAT).*** *For every class of components, we learn an augmentation $\Delta\mathbf{s}(t)$ of the current state $\mathbf{s}(t)$, taking the simulated constraint forces $\mathbf{f}_{\mathcal{C}}(t)$ and reference curves $\mathbf{r}(t)$ as additional inputs. Curved arrows represent skip connections.*

2.4 Transformer-based Augmentation

Our SAT is an instance of the *Temporal Fusion Transformer (TFT)* introduced by Lim *et al*. [2021]. The TFT was designed for a variety of forecasting tasks. All base modules required for an implementation are supported by Beitner *et al*. [2020]. In our setting, there is only a causality going forward in time (*i.e.*, the current state is only influenced by previous states). Moreover, unlike most forecasting tasks, our model consumes an initial state estimate and predicts an additive residual value. Our custom instance of the more general TFT is shown in Fig. 2.3.

The encoder of our SAT consumes the previous simulation states, together with the difference between measured and simulated states, $\Delta\bar{\mathbf{s}}$, or the previously predicted error $\mathbf{S}\Delta\mathbf{s}(t)$. This is motivated by the teacher forcing approach proposed by Williams and Zipser [1989]. At the initial time step, we assume the error to be zero or known. For electromechanical components, we can feed the network with measurements even during the evaluation phase. For rigid bodies, we feed the network with predictions from previ-

ous time steps or set them to zero. The decoder only consumes the current observable state.

Both the encoder and decoder transform the simulation states into a latent representation. This embedding has aggregated local information and the SAT continues with a multi-head attention layer [Vaswani et al., 2017]. The attention-weighted combination of the embeddings now encodes global information and is further processed in a final output layer which predicts a residual value for the current simulation state. For the backbone of the encoder/decoder, we experimented with different architectures, including dense fully-connected layers or the LSTM cells that are used in the TFT.

The model makes use of *Variable Selection Networks (VSN)* to assign varying importance to the input dimensions, suppressing irrelevant or disturbing fields in each prediction step.

Many hyperparameters can be chosen in our architecture. We can reliably estimate some of them (*e.g.*, hidden dimensionality or numbers of attention heads) using hyperparameter optimization frameworks [Akiba et al., 2019]. An interesting hyperparameter in our setting is the number of previous states or the length of the history that the model consumes in a single prediction step. In our experiments (Sec. 2.5), we show the attention profile of the TFT and how it improves prediction performance.

2.4.1 State Augmentation

To perform state augmentations, the model must handle fields with different scales and units. We apply standard normalization. For example, we define four global scale factors for state quantities that describe the position and orientation of a body and its linear and angular velocity. We then scale all coordinates of a particular 3-dimensional quantity with the same global scale factor.

For orientations, we rely on a 6-dimensional representation that is continuous and results in smaller errors for regression tasks [Zhou et al., 2019]. This representation is obtained by dropping the last column of a 3×3 rotation matrix. To recover the rotation matrix, we perform a Gram-Schmidt orthogonalization step on the first two columns.

2.4.2 Training

A supervised training procedure is used where the SAT minimizes an objective function between the predicted $\mathbf{S}\Delta\mathbf{s}$ and the measured error $\Delta\bar{\mathbf{s}}$ at each time step. To reduce the final sim-to-real gap of the simulation, we hence want to reduce an objective based on the absolute state values as, *e.g.*, the $\mathcal{L}_1$ loss.

As we are augmenting a simulation where physics must hold, we also propose an additional physics-informed term for position and velocity augmentations, where we penalize differences between the time derivative of the position and the velocity. The full physics-inspired loss is given as

$$\mathcal{L}_{PL1} := \mathcal{L}_1(\mathbf{S}\Delta\mathbf{s}, \Delta\bar{\mathbf{s}}) + \mathcal{L}_1\left(\frac{d}{dt}\boldsymbol{\theta}(\mathbf{S}\Delta\mathbf{s}), \dot{\boldsymbol{\theta}}(\mathbf{S}\Delta\mathbf{s})\right), \tag{2.3}$$

where $\boldsymbol{\theta}(\cdot)$ and $\dot{\boldsymbol{\theta}}(\cdot)$ extract the position and velocity from the state augmentation, respectively, and we use finite differences to approximate the time derivative of the position. We demonstrate later that the additional loss reduces overfitting (Sec. 2.5.4).

To compare two orientations $\mathbf{R}_1$ and $\mathbf{R}_2$, we rely on the geodesic distance

$$D(\mathbf{R}_1, \mathbf{R}_2) = \cos^{-1}\left((tr(\mathbf{R}_1\mathbf{R}_2^{-1}) - 1)/2\right). \tag{2.4}$$

2.5 Evaluation and Results

For the following experiments, we consider three robot configurations as shown in Fig. 2.4. KickBot is a small custom humanoid robot (height 438mm, mass 3.08kg, 12 DoFs), which we either attach to mechanical ground at the pelvis (KickBotA) or on one foot (KickBotB). These two configurations lead to significantly different forces and torques, and also show that our method can handle changes in contact configurations as would be seen for legged robots. DanceBot is a small biped (height 325mm, mass 2.58kg, 12 DoFs), with series-parallel kinematics, where most of the actuators are placed in the body, and we fix its feet to mechanical ground. The robots are driven with *Dynamixel XM430-W350-R* actuators and are 3D printed. The KickBot has motion capture markers attached to seven rigid bodies.

For the training data, we sampled 600 trajectories of 30s each, sampled at 250Hz with the method described in Sec. 2.3.3. The test data was collected from artist-created animations. The animations allow us to show generaliz-

ability over motions not sampled from the same underlying model used for the training data.

During the training of the SAT models, we never showed any data from KickBotB or DanceBot; the only training data comes from KickBotA. This makes our training-test split strong, and allows us to show generalizability over new, unseen robot configurations. The robots are built using the same actuators.

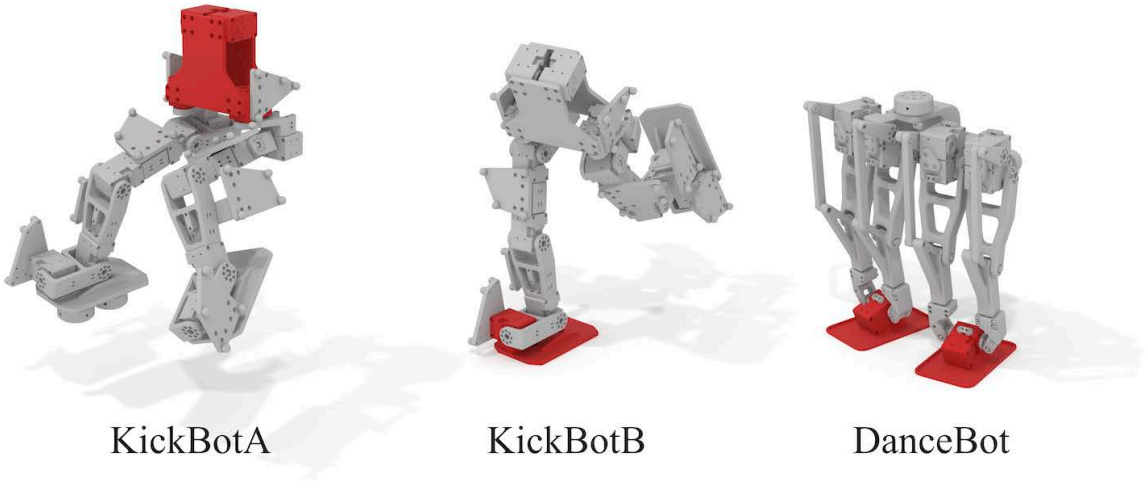


Figure 2.4: ***The robots.*** *KickBotA (attached at the pelvis), KickBotB (attached at the right foot), and DanceBot (attached at both feets). Fixed components are shown in red.*

2.5.1 Actuator Modeling

This section presents the augmentation capabilities of the SAT model on actuator states, which include position, velocity, and electric current. For this test, the SAT is solely trained with randomly sampled animations simulated and measured on KickBotA, and evaluated on unseen artist-created animations on KickBotB and DanceBot. Thus, we demonstrate that the model generalizes over different robots and over unseen animations. We compare our modular approach against a non-modular version of the SAT, where the augmentation takes the state of all actuators as input and learns to augment all states simultaneously.

In the first experiment, an instance of the modular and non-modular SAT learns to augment the actuators' position and velocity states simultaneously, with the physics-informed loss.

Fig. 2.5 shows bar plots summarizing the augmentation performance on the training robot (KickBotA) and the unseen robot (KickBotB) for position and

velocity, respectively. We evaluate the models on three metrics: the Concordance Correlation Coefficient (CCC) [Lawrence and Lin, 1989]; the Mean Absolute Error (MAE); and the max error. A higher CCC indicates that the state follows the up and down trend of the measurements more accurately. For the max error and MAE, we evaluate the deviation between the measurements and the simulated/augmented states per trajectory, and look at the error distribution over all test trajectories.

The simulation shows a max error of $>1^o$ on average, which the SAT state augmentation reduces to 0.6^o. The non-modular approach performs slightly better on the training robot (KickBotA), especially since the variance is smaller. However, it performs worse and has almost no improvement on the KickBotB. This suggests that the non-modular approach overfits the error patterns of the specific actuators that do not generalize to new acting forces and torques. Our proposed modular approach, however, achieves similar performance on the new configuration.

Fig. 2.7 shows an example of the state of an actuator over a window of 4s. Even during such a controlled motion with a velocity of 6^o/s, we observe a simulation gap. The augmented simulation matches the measured state for position and velocity more closely than the initial simulation. Note that the measurements are quantized, and we avoid pre-filtering in our training pipeline. Due to the physics-based loss, we see that the velocity augmentation follows the expected smooth profile although supervised on the quantized measurements.

We further show the augmentation capabilities on an unseen robot with a different topology (DanceBot, Fig. 2.5). The difference between the two unseen robots is the higher torque and complexity of the DanceBot, resulting in a more significant simulation error. Note that the non-modular approach is inherently topology-dependent by design, and can not be applied to a different robot. Our modular method can augment the actuator states and reduces the max simulation error from $\sim 3^o$ to $\sim 1^o$. In general, the state augmentation shows that the mean and variance of the simulation error can be reduced significantly for both unseen robots.

Besides the mechanical, the model can also learn to augment the electric states. We show the results of a modular SAT instance trained on predicting the error of the electric current of actuators in Fig. 2.6. The SAT increases the CCC from 0.25 to 0.75, showing that the model can capture the trends of the electric current more accurately.

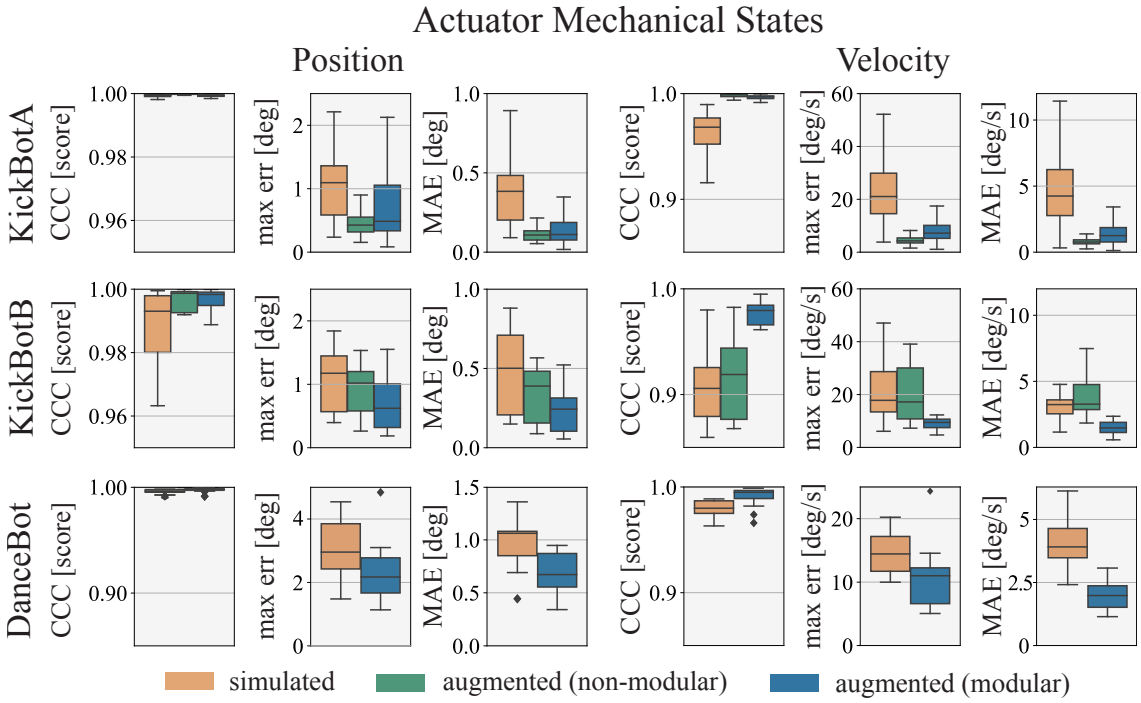


Figure 2.5: ***Actuator Augmentation Evaluation.*** *Position and velocity, for KickBotA and KickBotB non-modular vs. modular approach. And modular approach on unseen DanceBot.*

2.5.2 Rigid Body Modeling

We repeated similar experiments for the rigid bodies. We tracked multiple rigid bodies of the KickBot with a motion capture setup while performing the randomly generated trajectories. We train an instance of the SAT to predict augmentations for the 3D positions of the rigid bodies, and a second SAT on augmenting the 6D representation of the orientations. We show results for the KickBot in Fig. 2.8. For position, we measure the Euclidean distance, and for orientation, we evaluate the geodesic distance (Eq. 2.4).

The simulation has an average max position error of $\sim$17.5mm on KickBotA. Both the non-modular and modular approach can reduce the error by a factor $\sim$3.5, to $\sim$5mm. A similar effect is seen in the MAE.

The rigid body orientation error sees a similar reduction, with max error reduced from 7^o to $\sim 2.5^o$ and MAE reduced from 3.5^o to $< 1^o$. The non-modular approach performs similarly or better on the training robot but performs worse on a new robot configuration.

In general, the augmentation method performs better on actuators. The rea-

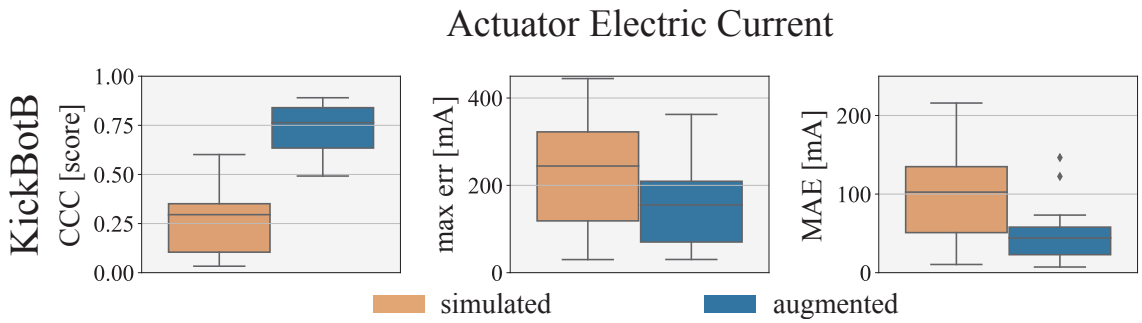


Figure 2.6: ***Actuator Augmentation Evaluation.*** *Electric current, for KickBotB.*

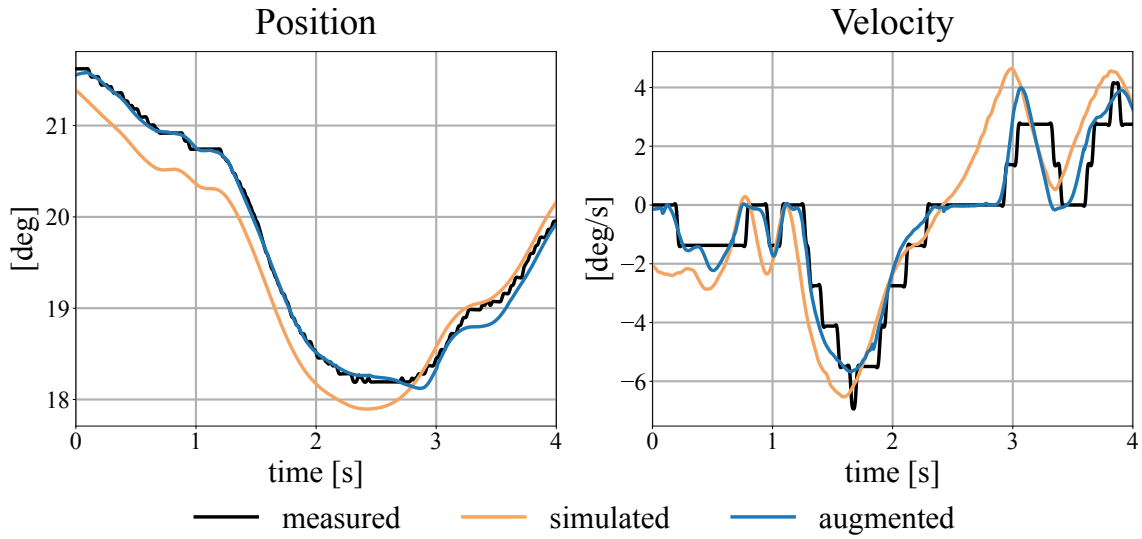


Figure 2.7: ***Actuator Augmentation Trajectories.*** *Illustrative example of time-varying trajectories. Measured (black), simulated (orange), augmented (blue).*

son for this is the higher dimensional space of the rigid bodies that are more difficult to sample densely.

2.5.3 Augmentation Limits

We further investigate the limitations of our augmentation. In a first experiment, we show augmentation limits when approaching actuator limits (Fig. 2.9). For this, we speed up the artist's animation by factors of $1 - 20$x. We see that the augmentation performance declines as we reach the actuator limits. At 10x speed-up, we are simulating motions that exceed the actuator limits; while the method can not achieve the same performance as before, it still improves the simulation states.

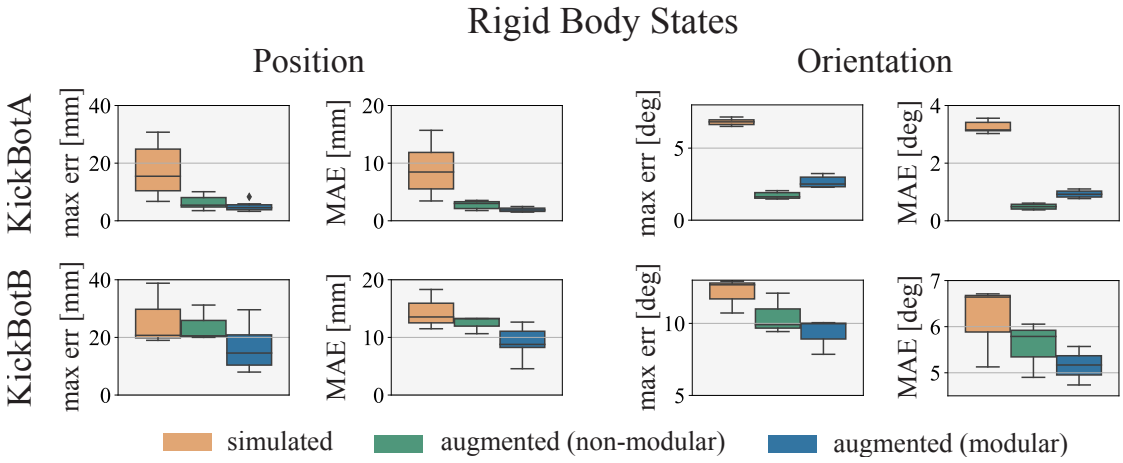


Figure 2.8: ***Rigid Body Augmentation Evaluation.*** *Position and orientation for KickBotA and KickBotB, non-modular vs. modular approach.*

Our rigid body augmentation is robust to model variations, *i.e.*, increasing sim-to-real gaps, as we demonstrate with the following experiments where we perturb simulation parameters with Gaussian noise of increasing magnitude. Concretely, we perturb each rigid body mass by $m_\epsilon = m\epsilon$, $\epsilon \sim N(1, \sigma)$, shift each rigid body center of mass by $COM_\epsilon = COM + \epsilon$, $\epsilon \sim N(0, \sigma)$, and constraint positions by $CTS_\epsilon = CTS + \epsilon$, $\epsilon \sim N(0, \sigma)$ for increasing values of σ up to σ_{max} as indicated in Fig. 2.10. For multiplicative noise, we ensure values remain non-negative by clipping to a small positive value. Note that we set σ_{max} to a large value (cf. Fig. 2.10), well beyond what could be expected as a sim-to-real gap, in order to stress-test our method.

As seen in Fig. 2.10, the rigid body augmentation is robust for a large perturbation and still achieves 50% improvement against the simulation for $\sigma_{max}/2$ and over 25% improvement for σ_{max}-perturbation. We also show the perturbation of the actuator's resistance parameter ($r_\epsilon = r\epsilon$, $\epsilon \sim N(1, \sigma)$), which shows a similar trend. Similar effects are observed when perturbing other actuator parameters. This shows that the augmentation is robust in a large neighborhood under uncertainty in simulation parameters and is evidence of generalization across different robot instances.

2.5.4 Ablation Studies

In the following, we evaluate the importance of the components of the SAT model as well as alternatives, give intuition into the attention mechanism, investigate different loss functions, and show how a physics-motivated term helps the model converge and stabilizes the training.

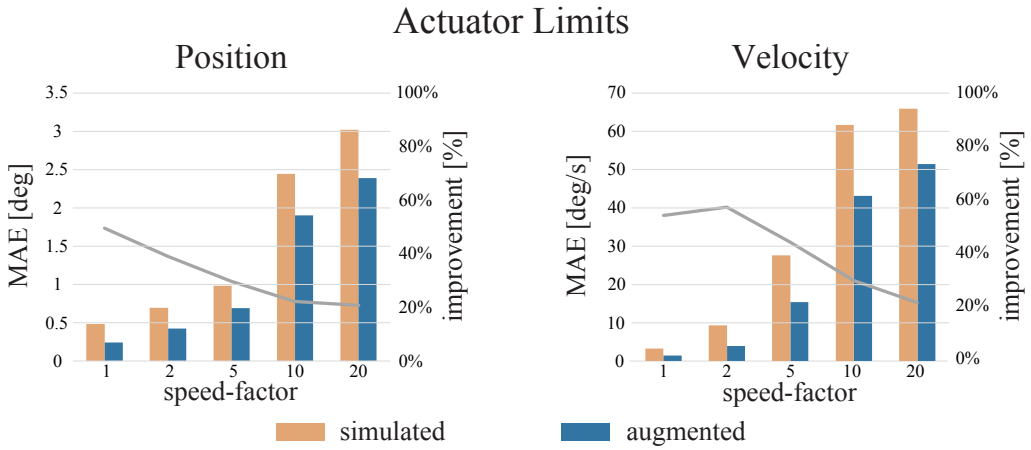


Figure 2.9: ***Augmentation Limits.*** *Position and velocity MAE of KickBotB on artist-designed motion for different speed-factors under and over the actuator limits. Gray lines show percentage improvement.*

Architecture

The final architecture combines recurrent neural layers with an attention-based module. To evaluate the importance of each component, we investigated the modules' performance in isolation and with alternative architectures as the encoder/decoder backbone. Each instance was trained to augment the position state of the actuators. All models saw the same training data, and were evaluated on unseen animations. The results are summarized in Table 2.1, where we report the MAE and Max error of the base simulation, and the augmentation performance of the different architectures. The LSTM model without attention results in an increased error, while the attention models can consistently improve the base simulation and reduce the MAE and Max error. A more complex encoder/decoder backbone has slightly better performance. To verify the results, we also train instances on augmenting position states for the rigid bodies of KickBotA, see Table 2.1. We see similar results here, where the attention models beat a pure LSTM model and reduce the mean and max error of the rigid body position.

In the following, we discuss the results for each architecture in more detail.

RNN We removed the attention part of the model and investigated the output of the Variable Selection Network (VSN) [Lim et al., 2021] followed by the LSTM layer. The VSN acts as a pre-filtering of the input fields, avoiding the disturbance of the limited LSTM cells caused by less important or irrelevant fields. This pure LSTM approach stagnates on long animation sequences, leading to an increased max- and mean-error of the simulation, on

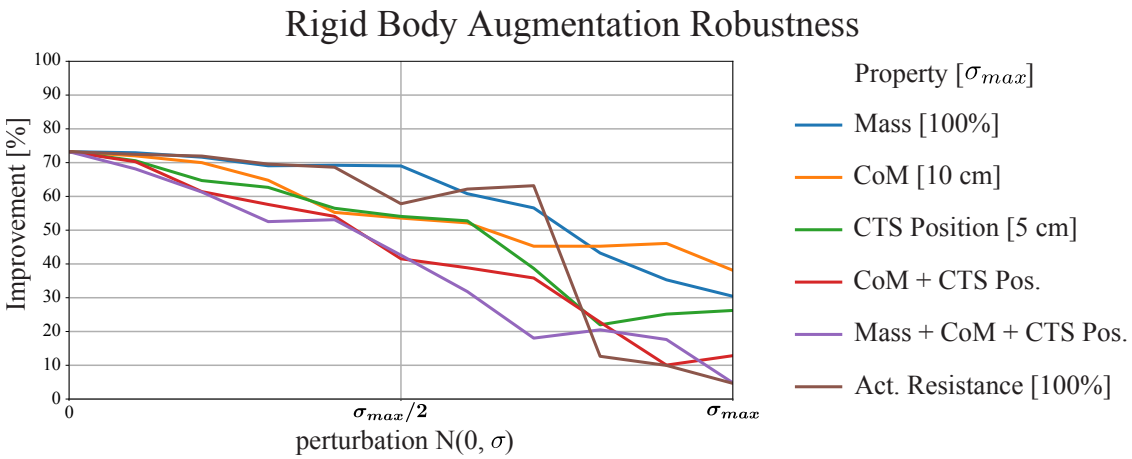


Figure 2.10: ***Augmentation Robustness.*** *Augmentation around sampled rigid bodies is robust for large perturbations in Mass properties, shifting center of mass (CoM), moving constraint positions (CTS Pos.) and combinations. We also see robust behavior when changing actuator parameters like the resistance.*

Table 2.1: ***Architecture Evaluation on KickBotA.*** *Lowest values in bold. Act.: Actuators, Rbs: Rigid Bodies, Attn.Dense: Attention + Dense.*

	Simulation		**LSTM**		**Attention**		**Attn.Dense**		**SAT**	
	MAE	Max	MAE	Max	MAE	Max	MAE	Max	MAE	Max
Act. [deg]	0.48	1.10	1.50	2.80	0.26	0.82	0.26	0.76	**0.24**	**0.71**
Rbs. [mm]	8.72	17.52	2.78	7.48	**1.87**	5.57	2.43	6.12	1.90	**5.03**

average by a factor of 2. We observe that the stagnation of the model is slower on rigid bodies, and the augmentation beats the base simulation on the 30s-long animations. Besides a lower error accumulation, this also suggests that there are some simpler error patterns for rigid bodies where an RNN can already improve. The MAE can be reduced by 70% from 8.7mm to 2.8mm.

Attention We investigate whether attention can improve the results by removing the LSTM and using the multi-head attention layer with its final fully connected output module. In this configuration, the model applies attention to the output of the VSN module and has an unlimited perceptive field without information losses. The transformer aggregates the past and current simulation states together, and a final dense output layer predicts the residual of the position state. This very simple model significantly reduces the sim-to-real gap. The average improvement on the unseen robot is 45% for the mean error and 25% for the max error. The attention model

also further improves the results on the rigid bodies by reducing the MAE by 80%. The aggregation with the attention layer is powerful enough for a simple dense output layer to outperform the RNN model consistently and significantly over a larger prediction period. This shows the attention layer to be crucial in the architecture and sufficient for achieving a good augmentation. However, we further investigate the composition of both architecture types as proposed in the initial TFT model and alternatives.

Encoder/Decoder Starting from the simple transformer model above, we investigated different extensions of the encoder/decoder. The first version *Attention Only* is the previously described model with no additional layers attending directly to the output of the VSN. *Attention + Dense* has two additional Dense ReLU layers after the VSN. Finally, *Attention + LSTM* describes the encoder/decoder architecture proposed in the TFT with an LSTM layer that is recurrently applied over the history sequence. The additional components in the encoder/decoder lead to slightly better results. Notably, the Max error can be reduced by a further 5% by the dense layers and 10% by the LSTM layer on average for both robots. We, therefore, further investigate their effects on the attention layer.

Fig. 2.11 shows the normalized average attention of each architecture over the input states in percentage. We see that the *Attention Only* and *Attention + Dense* models assign over 10% of their attention to the single previous history state. This is significantly more than the *Attention + LSTM* model with only 4.5%. The *Attention + Dense* instance assigns similar attention over the last 10 states. In the cumulative sum of the attention plot, shown in Fig. 2.11 (right), we observe that using no additional layers in the encoder assigns only 27% of the attention to states that are 20 or more steps in the past. Using the dense layers increases this attention to 40%, but assigns it almost equally along the states. The model with LSTM-cells shows a more gradual decrease of attention for past states.

Thus, we refer to this final attention model with an LSTM encoder/decoder as the State Augmentation Transformer (SAT). In contrast to the TFT, the SAT consumes no future or static covariates and acts as a residual network predicting the correction of the current simulation state.

Physics-Informed Loss

We evaluated the convergence of the SAT when trained on different loss functions. We train an SAT model to augment the actuators' position and

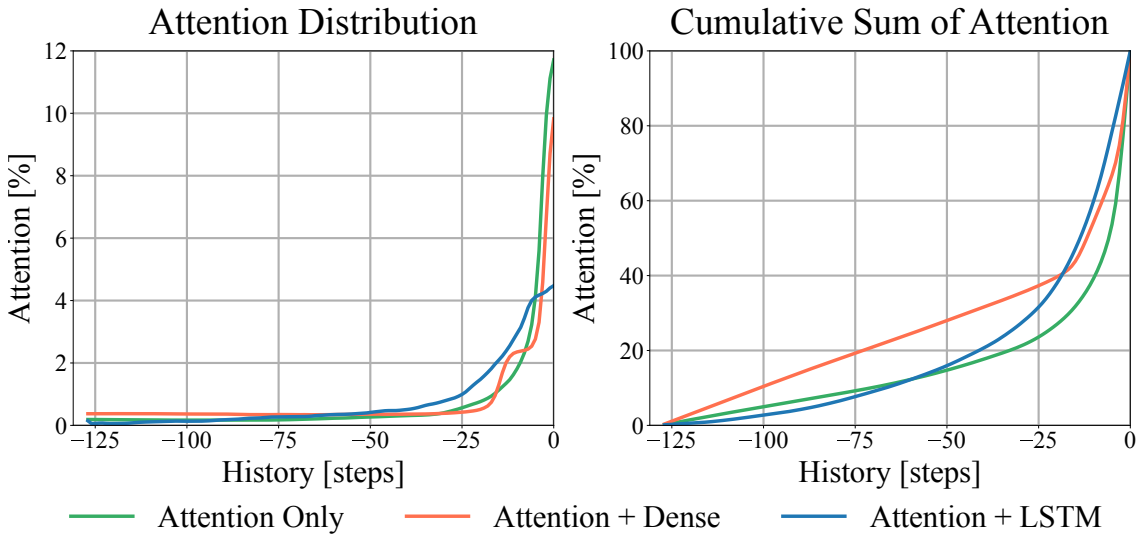


Figure 2.11: ***Attention over 128 history states.*** *The attention profile for the three encoder/decoder variations. Left: Normalized average attention per history step. Right: Cumulative sum of the attention distribution.*

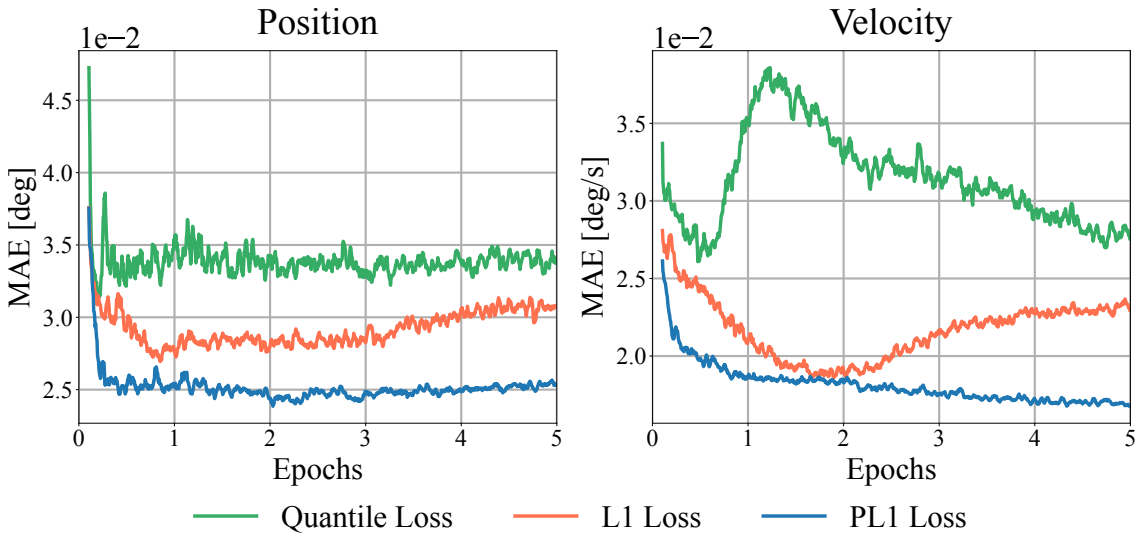


Figure 2.12: ***Validation Loss.*** *We trained the SAT model to simultaneously augment the position and velocity state. Here we show the Mean Absolute Error (MAE) on an unseen motion while training with three different objectives.*

velocity state simultaneously. Fig. 2.12 evaluates the MAE of an unseen animation during the training process. We use the Quantile, $\mathcal{L}_1$, and the Physical-Informed $\mathcal{L}_1$ loss ($\mathcal{L}_{PL1}$). In contrast to usual forecasting tasks, we observe that the Quantile loss does not fit our problem. Its statistical nature does not concentrate on reducing the absolute error between the simulation and the measurements. We observe that the MAE of the position augmentation saturates quickly and increases temporarily for the velocity. With the $\mathcal{L}_1$ loss, we see a significant decrease in the MAE. The model's predictive power increases, and we learn more general applicable patterns from the training data. Over the training time, however, we observe an overfitting effect that leads to increasing position and velocity errors, which is a known issue with neural networks [Zhang et al., 2021]. The $\mathcal{L}_{PL1}$ loss shows a lower MAE. This loss acts as a filtering process that reduces the effect of noise and mitigates overfitting.

2.6 Concluding Remarks

Contributions. We have devised a transformer-based augmentation of several robot building blocks, and demonstrated that we can learn actuator augmentations that generalize well to other robots, and rigid body augmentations that are robust under uncertainty in modeling parameters. We have presented augmentations for positions, velocities, orientations, and electric currents of components. Moreover, we have evaluated the attention mechanism and argued for the fused LSTM-Transformer architecture. Additionally, we have presented an idea of introducing physical information into the loss function and shown this regularization to positively affect model convergence.

Limitations. One limitation of the augmentation is that the method has no mechanism to keep constraint violations low. While this allows the augmentation to move out of the constrained simulation space and more accurately match the measured data, it could also lead to unwanted constraint violations for scenarios with insufficient training data. While we only observed relatively small constraint violations in our examples, additional loss terms that minimize such a coupled error metric could be an interesting direction to explore.

While we have demonstrated generalization of actuator augmentations across different robots, our rigid body augmentation does not generalize to new robots. Because we observe good generalization in a neighborhood of

a moving rigid body, we believe that a generalization across robots made of different rigid components could be possible with an extensive sampling of rigid body properties and behaviors, which we leave as future work.

Soft components are typically simulated using finite elements. As the number of elements vary per component, our modular approach is not directly applicable. The augmentation of general soft bodies with a single transformer is an interesting direction for future research.

CHAPTER 3

Sequential Data Representation

Positional encoding is an essential component in transformer models, introduced in the seminal work by Vaswani *et al*. [2017]. It infuses positional information into input tokens to help transformers learn position-agnostic token embeddings. Positional encoding works by (1) pre-assigning sinusoids of different frequencies and phases to every position an input token can take on in a sequence, and (2) by adding this sinusoid to the token embedding that appears at the corresponding position in the sequence. Injecting a token with positional information, also referred to as *absolute* position encoding in later work, has evolved into several variants that address shortcomings and improve generalization. For example, several works have shown that absolute position encoding limits the ability of transformers to handle longer sequences at inference time and proposed *relative* position encoding schemes where a fixed or learned bias is added to the attention matrix [Raffel et al., 2020; Press et al., 2022; Ke et al., 2021]. Irrespective of their exact arrangement, today's state-of-the-art transformer architectures employ a combination of absolute and relative position encoding schemes [Press et al., 2022; Su et al., 2023; Ruoss et al., 2023].

Positional encoding assumes that token embeddings represent elemental data in a collection, *e.g*., individual words in a sentence, images in a video, or poses in an animation, and that an additional notion of position is required to model a collection of such elements, such as sequences of words, images, or animation frames. This thought process conceptually decouples

This chapter is based on the publication *"Spline-Based Transformers"*, Prashanth Chandran*, Agon Serifi*, Markus Gross, and Moritz Bächer [Chandran et al., 2024]. A supplemental video is available at `https://youtu.be/AzolLlIbKhg`.

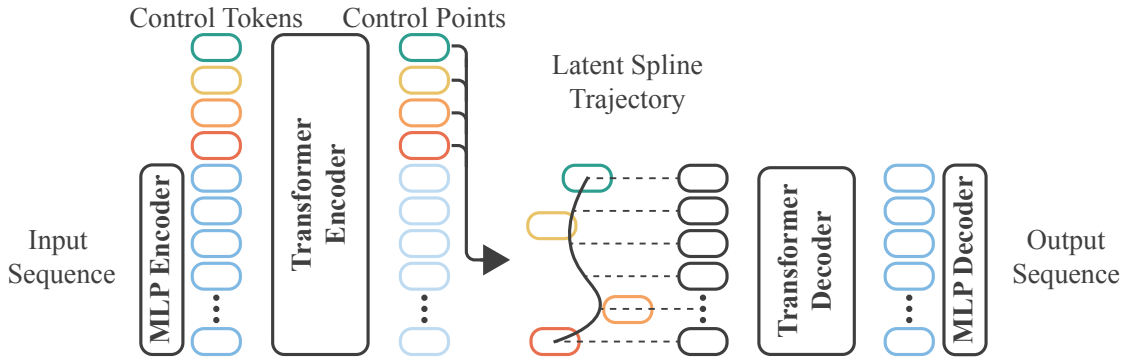


Figure 3.1: ***Spline-based Transformers.*** *Our Spline-based Transformers encode an input sequence, together with learnable control tokens, into a trajectory in latent space defined by the latent control points of a spline curve.*

elemental and collective datatypes and forces that separate representations for the elements and the collection as a whole are learned. This becomes even more evident when we consider that most existing architectures that learn compact neural representations for collections do not leverage the fact that individual elements, traversed in a particular order, make up a collection.

In this chapter, we argue that learned neural representations for elemental and collective datatypes do not have to be decoupled from one another. Instead, they can be effectively represented in a single, shared latent space. At the heart of our approach is the idea that a collection can be represented by learning to traverse a trajectory in the latent space of elemental data. Inspired by animation workflows where splines are commonly used to describe a temporal sequence of poses, we introduce a new class of transformer models based on splines that we call *Spline-based Transformers*. They do not require absolute position encoding.

At a high level, our approach uses a transformer-based encoder with additional learned control tokens to reduce an input collection of elements to a fixed number of latent control points $\in \mathbb{R}^d$. These control points are interpreted as the control points of a d-dimensional spline in latent space, representing a continuous latent space trajectory. The trajectory encapsulates the fundamental characteristics of the elements constituting the input collection. Uniformly sampling and processing the trajectory through the transformer-based decoder reconstructs the original input sequence. Our Spline-based Transformers require no sinusoidal positional encoding and, therefore, completely circumvent the downsides of absolute position encoding, including poor extrapolation and overfitting. A conceptual overview of our approach is illustrated in Fig. 3.1.

We demonstrate the superior performance of our Spline-based Transformers over transformers with conventional positional encoding on several datasets and applications, including synthetic data (Subsec. 3.3.1), images (Subsec. 3.3.2), animation data (Subsec. 3.3.3), and in representing challenging geometry like hair strands (Subsec. 3.3.4). Additionally, our Spline-based Transformers allow users to manipulate a given collection by directly interacting with corresponding latent controls, thereby introducing a new means of interacting with this architecture. With transformers gaining significant attention in recent years as general-purpose architectures [Dosovitskiy et al., 2021; Radford et al., 2023; Petrovich et al., 2021; Chandran et al., 2022b; Aneja et al., 2023; Peebles and Xie, 2023], we believe that our simple yet effective approach has the potential to be leveraged across multiple disciplines for a wide variety of tasks.

Succinctly, our contributions are:

- We introduce Spline-based Transformers; Transformer models that use a spline-based latent space to encode temporal information without requiring additional positional encoding.
- We show that simple control mechanisms to manipulate the latent space are automatically learned by our models and allow for rapid manipulation of the output sequence.
- We demonstrate superior performance of Spline-based Transformers over Transformers with positional encoding on a variety of data modalities, including synthetic data, images, 3D shapes, and motion datasets.

3.1 Related Work

Before we discuss related work, we would like to clarify our use of the term positional encoding. The term positional encoding is widely used in the literature of coordinate-based neural networks [Mildenhall et al., 2021; Park et al., 2019a] to refer to a frequency encoding scheme for improvement of network training [Tancik et al., 2020] wherein a low dimensional input (such as a 3D position) is mapped to a higher dimension using a collection of sinusoids of different frequencies. In this chapter, we refer to positional encoding as a mechanism to *introduce* positional information into inputs and outputs that are otherwise devoid of any positional information, as is common in the literature on transformers.

Transformers [Vaswani et al., 2017] were introduced as an alternative to

traditional sequence models such as RNNs [Rumelhart et al., 1986] and LSTMs [Hochreiter and Schmidhuber, 1997]. While they initially showed remarkable performance on language tasks, their effectiveness as a general purpose neural architecture led to their quick adoption as foundational image models [Dosovitskiy et al., 2021; Liu et al., 2021; Liu et al., 2022], in speech recognition [Radford et al., 2023], 3D and 4D modeling [Chandran et al., 2022b; Aneja et al., 2023; Petrovich et al., 2021; Chandran et al., 2022a], and more recently as backbone architectures for diffusion models [Peebles and Xie, 2023].

The original transformers [Vaswani et al., 2017] relied on absolute position encoding with sinusoids to inject positional and temporal information into input tokens. Soon after, researchers identified sequence length extrapolation and overfitting as limitations of absolute positional encoding and introduced several extensions to combat them. Su *et al*. proposed *RoPE* [Su et al., 2023], where absolute positions are encoded as a rotation matrix and relative token positions are explicitly taken into account during attention computations for better performance and generalization. Raffel *et al*. introduced the T5 transformer [Raffel et al., 2020] where a learned bias that depends on the relative distance between tokens is added to the attention matrix. They achieved a performance boost on a variety of natural language tasks. Press *et al*. proposed *ALiBi* [Press et al., 2022] and showed that a fixed bias with a predetermined slope that depends on the attention head can improve the performance for unseen sequence lengths. An extension to ALiBi [Al-Khateeb et al., 2023], which applied ideas from RoPE, further improved the performance of ALiBi in language tasks. To specifically tackle the extrapolation problem, Ruoss *et al*. proposed positional encoding with a randomized ordering of sinusoids [Ruoss et al., 2023] to account for longer test positions by augmenting the training distribution. A more recent and highly relevant finding in the context of this chapter is described by Kazemnejad *et al*. [2023]. They showed that transformers with no positional encoding (NoPE) outperform most commonly used forms of position encoding in *decoder* only tasks.

However, for transformers used in learning condensed latent spaces of sequential data [Radford et al., 2021; Tevet et al., 2022; Duan et al., 2021; Petrovich et al., 2021; Chandran et al., 2022a; Chandran et al., 2022b], almost all current methods make use of a transformer autoencoder with relative position encoding [Press et al., 2022; Raffel et al., 2020] and an additional [CLS] (short for classification) token [Devlin et al., 2018] as input. In these applications, the transformer encoder aggregates information from the input data tokens into the [CLS] token, which is then interpreted as a condensed latent descriptor of the input sequence at the encoder's output. This latent descriptor token is appropriately position-encoded and passed through a

transformer decoder to reconstruct the input. For such scenarios, the use of no positional encoding (NoPE) is not a viable solution as it reduces to an n-fold duplication of the latent descriptor, therefore passing an identical or static latent sequence to the decoder. Without any variation in its input or absolute positional encoding, the decoder fails to reconstruct the original input sequence.

Our Spline-based Transformers present a new approach to learning such condensed latent spaces for sequential data using a transformer autoencoder that does not require a positional encoding. In addition to providing significant performance benefits, our approach provides a novel control mechanism to navigate the latent spaces without any additional complexity.

3.2 Spline-based Transformers

The core of our architecture is constructed around a transformer autoencoder model, which incorporates a latent space between the encoder and decoder components. In the following, we first introduce the theory behind the modifications that result in our Spline-based Transformers and later describe the architectural details of the new transformer autoencoder. See Fig. 3.1 for an illustration.

3.2.1 Background: Splines

Splines have seen widespread use in function approximation, computer-aided design, and the specification and editing of animation curves in computer graphics. They provide a means to define a curve or a trajectory with a discrete set of control points and have many desirable properties. Adjustments to control points have only a local effect, and the degree of the polynomial basis provides users with control over the smoothness of a curve.

While our modeling is agnostic to the specific spline representation, we use B-Splines in our current transformer model as they provide a good trade-off between ease of implementation and fine-grained control over shape and smoothness. A B-Spline curve is a linear combination of control points, $\mathbf{p}_i$, and basis functions, $N_{i,k}(t)$,

$$\mathbf{s}(t) = \sum_{i=0}^{n} N_{i,k}(t)\mathbf{p}_i \quad \text{for} \quad t \in [t_{k-1}, t_{n+1}], \tag{3.1}$$

and describes a piecewise polynomial curve where each segment has degree

k [Farin, 2001]. The smoothness at the interface of pairs of segments is determined by the knot vector

$$T = (t_0, t_1, \ldots, t_{k-1}, t_k, t_{k+1}, \ldots, t_{n-1}, t_n, t_{n+1}, \ldots, t_{n+d}). \tag{3.2}$$

Note that a B-Spline curve does, in general, *not* pass through the two end control points. Only if a knot has multiplicity $k-1$, the corresponding control point will lie on the curve, reducing the continuity at that point to $\mathcal{C}^0$. If we increase the multiplicity of a knot to k, the curve is $\mathcal{C}^{-1}$ and therefore discontinuous. In more general terms, a knot with multiplicity m results in a curve that is $k-m-1$-differentiable, and hence C^{k-m-1}, at the knot. We can always normalize the time interval so that $t \in [0, 1]$.

Splines have many desirable properties, notably:

- *Local support:* A knot span, $t_i \leq t \leq t_{i+1}$, is only affected by k control points, and a control point only has an effect on k spans. Adjustments to a control point, $\mathbf{p}_i$, have an effect on the curve between t_i and t_{i+k}.
- *Smoothness:* If the multiplicity of a knot is zero, a B-Spline curve is $\mathcal{C}^{k-1}$ and $k-1$-differentiable. To increase the smoothness of the curve, we can always increase the degree of the polynomial basis.
- *Numerical Stability:* The theory behind B-Splines is well-understood, and numerically stable and efficient algorithms exist to evaluate them [Farin, 2001].

3.2.2 Network Architecture

Typically, the encoder of a transformer autoencoder reduces an input sequence of tokens into a single latent code. However, because transformers are sequence-to-sequence architectures, they require additional pooling mechanisms to condense information from the entire input sequence into a single latent token [Devlin et al., 2018; Touvron et al., 2021]. This is usually accomplished by concatenating an additional learned token to the input sequence, and by using only the latent representation of this token as input to subsequent neural networks (*e.g.*, a decoder) or by directly using it in a training objective [Radford et al., 2021]. The other outputs of the transformers are discarded. In classification tasks, the learned token is often referred to as the [CLS] token [Devlin et al., 2018]. Finally, in order to decode the latent [CLS] token into an output sequence, it is duplicated, positional encoded appro-

priately, and passed through a transformer decoder that predicts the output sequence.

Instead of only appending a single [CLS] token to the input, Spline-based Transformers append a collection of ordered control tokens to the input sequence. Specifically, Spline-based Transformers append $n+1$ control tokens to the input sequence to obtain $n+1$ control points, $\mathbf{p}_i$, that will be used to evaluate a latent spline at the output of the encoder with polynomial basis of order k. Latent codes corresponding to each output token are produced by evaluating the spline at the token's position according to Eq. 3.1.

The resulting trajectory, $\mathbf{s}(t)$, in latent space, has several advantages compared to a traditional positional encoding. First, the latent code is not perturbed by positional information, meaning the decoder does not need to learn to distinguish between positional and contextual information. Second, when using sinusoidals to encode the position of tokens, the contextual part of the token remains fixed and therefore provides a form of redundancy; our latent spline trajectories encode the temporal information implicitly, *e.g.*, they can traverse the latent space faster in certain points and slower in others, making better use of the latent space. In Fig. 3.2, we show an overview of how our spline-based latent trajectories are derived from the control points and how they differ from commonly used schemes like ALiBi [Press et al., 2022].

Architecture Details. As seen in Fig. 3.1, an input sequence is encoded using an MLP that is shared across input tokens, leading to an embedded sequence. Learnable control tokens are concatenated to the embedded sequence and sent through a seq2seq transformer encoder block. We add a linear layer after the last transformer encoder block to map the encoded tokens to the latent space dimension d. The exact number of evaluations of the spline depends on the number of output tokens expected at the decoder's output. Our transformer decoder uses the same structure as our encoder. The encoder and decoder have n-layers, each layer has h heads, and c feature dimensions. In every layer of the transformer, an ALiBi attention bias is added. Each layer, except the last MLP of the decoder, uses the GeLU activation [Hendrycks and Gimpel, 2016]. While our transformer blocks follow the structure of the T5 transformer model [Raffel et al., 2020], any transformer block could be used in combination with the spline-based latent space. Depending on the complexity of the data type, we use transformer blocks of varying feature dimensions and capacities. For training our Spline-based Transformer autoencoder, we use the RAdam optimizer [Liu et

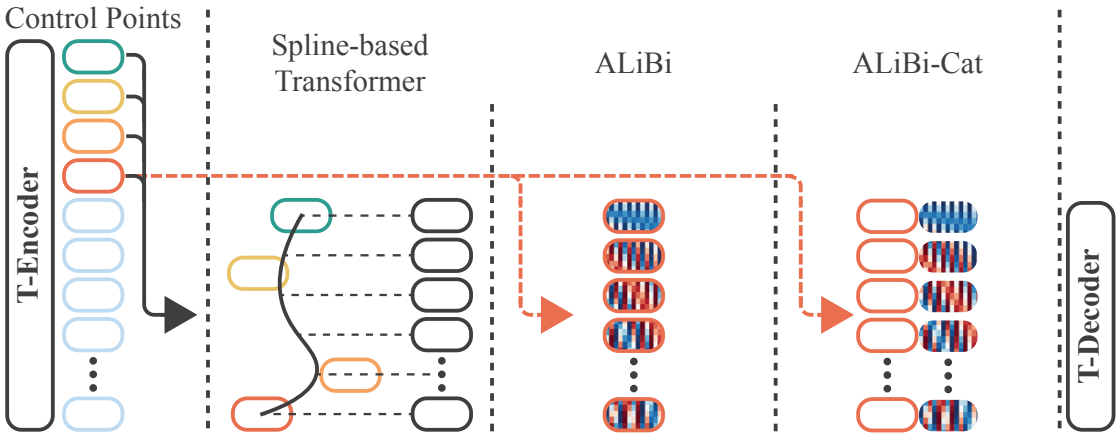


Figure 3.2: ***Variations of Latent Spaces.*** *Our Spline-based Transformers use multiple control points to evaluate a latent B-Spline and to create a d-dimensional trajectory in the model's latent space. In our experiments, cubic Bézier curves are used, which require four control points. On the other hand, ALiBi duplicates a single control point and adds positional information to the duplicated points, while the positional information is concatenated to the duplicated control point in ALiBi-Cat. The latent representations are ordered from top to bottom in time. While the spline-based space encodes temporal information implicitly, the other methods incorporate it explicitly using sinusoidal curves of varying frequencies.*

al., 2020] with a cosine annealing learning rate scheduler [Loshchilov and Hutter, 2017].

3.3 Evaluation and Results

We now present experiments on a number of datasets to demonstrate the effectiveness of the proposed Spline-based Transformers when applied to multiple modalities of sequential data. We specifically compare our method against ALiBi [Press et al., 2022], a state-of-the-art transformer model that uses a combination of both absolute and relative positional encoding. Because ALiBi adds sinusoids to the input token embedding, which could create an ambiguity between the token's content and position for the transformer decoder to disentangle, we also compare against a variation of ALiBi, where the sinusoids are concatenated with the token embedding, effectively doubling the size of the transformer decoder blocks. We refer to this concatenated variation as ALiBi-Cat. Fig. 3.2 illustrates the differences between our spline-based latent space and the two baselines; ALiBi uses a single control point and adds positional information on top to create a latent sequence

[Chandran et al., 2022a; Daněček et al., 2023], while ALiBi-Cat concatenates the control point and the positional information instead.

For our experiments, we parameterize the latent space between the encoder and decoder blocks with cubic Bézier curves, with four control points per segment. Bézier curves are one instance of the B-Spline family, and provide sufficient smoothness for the applications we have studied so far. We uniformly sample the latent spline trajectory in the range $t \in [0, 1]$. We summarize the network parameters for each experiment in our supplemental material.

3.3.1 Synthetic Datasets

We first evaluate our Spline-based Transformer, ALiBi, and ALiBi-Cat in representing parametric 2D curves that have a known latent space size. For this task, we use three different parametric curve families: (1) Lissajous ($d = 3$), (2) Hypotrochoids ($d = 4$), (3) Bézier curves (with $d = 2$, and $d = 64$). For each curve type, we create three different transformer autoencoders for the Spline-based Transformer, ALiBi, and ALiBi-Cat, respectively. As seen in Fig. 3.2, the network architectures for the different autoencoders are identical, with the only difference being the mechanism used to derive latent token trajectories. The dimensionality of latent token embedding is decided based on the known latent space of the curve family. We train the three transformer autoencoders independently on each curve family. For training, we randomly sample curve parameters according to the parameterization of the curve in a pre-determined domain. Using the sampled parameters, we evaluate the curve to create a sequence of 256 2D tokens that contain the (x, y) coordinates of the curve, which are then fed as input to the transformer encoder. The three transformer autoencoders are trained end-to-end using a simple L2 reconstruction objective. In Table 3.1, we show the reconstruction error of each of the transformer models when presented with 10000 unseen curves from the family it was trained on. Our Spline-based Transformer outperforms ALiBi and ALiBi-Cat, especially on low dimensional latent spaces. Some qualitative comparisons are shown in Fig. 3.3.

Table 3.1: ***Synthetic Curves Reconstruction.*** *Average reconstruction error (MSE) of 10000 test curves in 2D*

Method	Lissajous (3D)	Hypotrochoids (4D)	Bézier (2D)	Bézier (64D)
ALiBi	1e-4	2e-3	1.76e-2	3.88e-3
ALiBi-Cat	8e-4	5.3e-3	1.78e-2	3.89e-3
Spline (Ours)	**3e-5**	**1.4e-3**	**2e-6**	**3.87e-3**

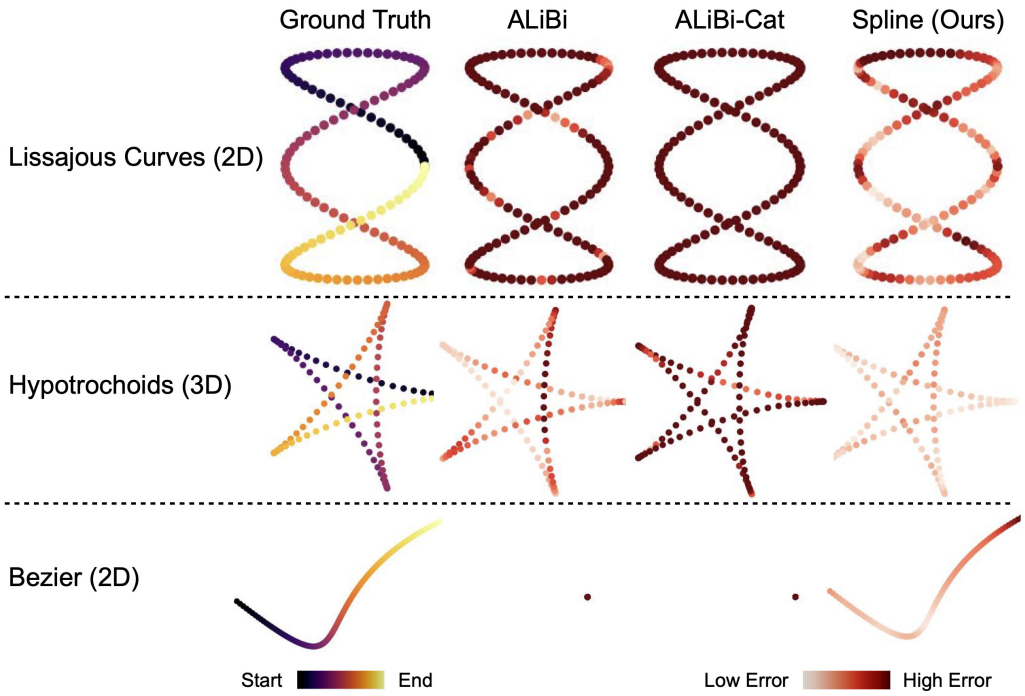
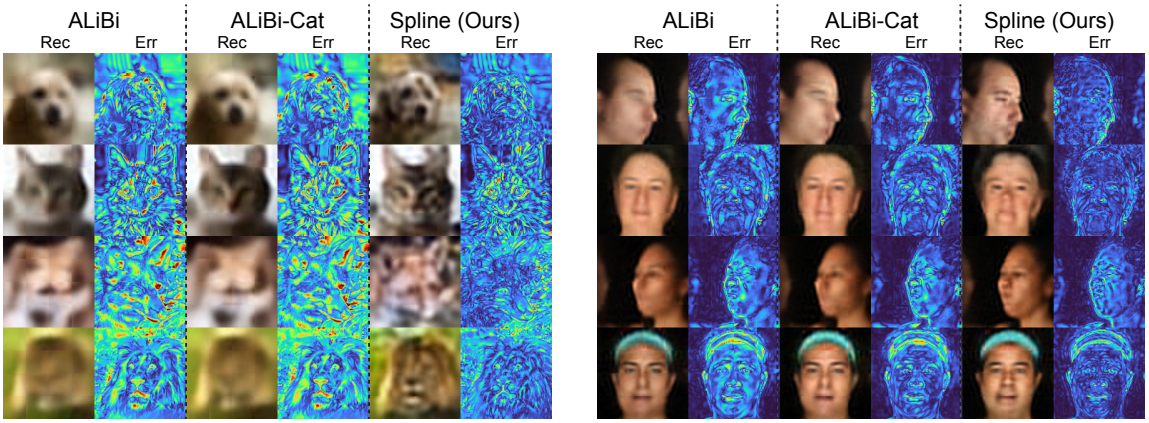


Figure 3.3: ***Visual Synthetic Curves Reconstruction.*** *Our Spline-based Transformer can successfully reconstruct curves of different families with consistently better performance than ALiBi and ALiBi-Cat. In certain scenarios (third row), reconstructions from ALiBi and ALiBi-Cat can collapse to a single point, while our Spline-based Transformer successfully manages to recover the input curve.*

3.3.2 Images

We continue by showing the effectiveness of Spline-based Transformers in reconstructing real image datasets. For the following experiment, we divide an image into distinct non-overlapping patches to create a sequence of patches similar to (Masked) Vision Transformers [Dosovitskiy et al., 2021; He et al., 2022]. The sequence of 2D image patches is then the input to our transformer autoencoder. We replace the MLP-Encoder in Fig. 3.1 with a CNN-Encoder that maps each patch of size $(PS, PS, 3)$ to a d-dimensional latent token $(1, d)$. An image (H, W) is therefore represented with a sequence of size $(HW/PS^2, d)$. The rest of the transformer autoencoder remains identical to what was described above. We train the transformer autoencoder using a simple L2 reconstruction loss to recover the input image from the patch sequence and compare the performance of the Spline-based Transformer against ALiBi and ALiBi-Cat.

We present results on three different image datasets: CIFAR-10 [Krizhevsky



(a) *AFHQ - 64D* **(b)** *Faces - 64D*

Figure 3.4: ***Visual Image Reconstruction.*** *Comparison between ALiBi, ALiBi-Cat, and Spline (ours) using a 64D bottleneck on AFHQ and Faces.*

and Hinton, 2009] (32x32), AFHQ [Choi et al., 2020] (128x128), and a dataset containing facial images [Chandran et al., 2020] (128x128). For each dataset and method, we train three transformers with three different latent sizes: 32D, 64D, and 128D. Table 3.2 summarizes the results. The spline-based latent space significantly outperforms the baselines by a factor of 2. Fig. 3.4 shows examples of the reconstructed images and their corresponding error maps; the spline-based latent space results in sharper and more detailed images. We observe that the performance improvements are larger for lower dimensional latent spaces. More results are reported in our supplementary.

Table 3.2: ***Image Reconstruction.*** *Comparison across different datasets and bottleneck dimensions.* ***Bold*** *indicates the best overall performance, and underline the best in each category. Performance is measured in Mean Squared Error.*

	CIFAR-10			AFHQ			Faces		
Method	32D	64D	128D	32D	64D	128D	32D	64D	128D
ALiBi	0.266	0.178	0.107	0.064	0.050	0.038	10.65e-3	8.56e-3	6.71e-3
ALiBi-Cat	0.264	0.174	0.108	0.064	0.049	0.038	10.87e-3	8.56e-3	7.14e-3
Spline (Ours)	0.107	0.056	**0.042**	0.038	0.030	**0.025**	6.77e-3	5.27e-3	**4.52e-3**

3.3.3 Animation

A commonly encountered modality of sequential data is 3D animation, with interest in using transformers for learning motion manifolds garnering significant attention in recent years [Petrovich et al., 2021; Chandran et al.,

2022a; Aneja et al., 2023; Daněček et al., 2023]. A Spline-based Transformer, when used to represent 4D data, like a sequence of 3D meshes from a facial animation or a sequence of joint poses describing human motion, can lead to notable performance benefits.

Faces. We compare the performance of the Spline-based Transformer autoencoder against ALiBi, and ALiBi-Cat, training the three models on a database of 3D facial animations [Chandran et al., 2022a]. Each animation is represented by a sequence of registered 3D meshes. We decimate them to meshes with around 5000 vertices, and flatten the vertices to a vector. A flattened animation sequence is thereafter split into windows of size 30 ($\sim$1 second of animation) and used to train three variations of the transformer autoencoder. In Table 3.3, we show the reconstruction error on six performances from three unseen identities. Irrespective of the dimensionality of the latent space, the Spline-based Transformer outperforms both ALiBi and ALiBi-Cat. A qualitative visualization of a reconstructed test performance is shown in Fig. 3.5 for the 64D Spline-based Transformer model. We note that the ALiBi transformer decoder used in these experiments is conceptually similar to the ones used in previous works [Petrovich et al., 2021; Chandran et al., 2022a; Daněček et al., 2023; Aneja et al., 2023], indicating that Spline-based Transformers could lead to improved performance in several downstream tasks.

Table 3.3: ***Face Performance Reconstruction.*** *Comparison across different latent dimensions.* ***Bold*** *indicates the best overall performance, and* <u>underline</u> *the best in each category. Performance is measured in Mean Squared Error.*

Method	32D	64D	128D	256D
ALiBi	1.58	1.55	1.48	1.54
ALiBi-Cat	1.60	1.54	1.53	1.50
Spline (Ours)	<u>1.43</u>	**1.35**	<u>1.47</u>	<u>1.47</u>

Full-Body Motion. We evaluate our method on the full-body human motion dataset HumanML3D [Guo et al., 2022], a combination of the HumanAct12 [Guo et al., 2020] and AMASS dataset [Mahmood et al., 2019]. In Table 3.4, we report the mean squared reconstruction error of per-frame joint positions measured in degrees. Spline-based Transformers show reconstruction improvements of at least a factor two, reducing the mean joint error of the smallest model (16D) from $\sim 0.4^\circ$ to $\sim 0.2^\circ$, and up to $\sim 0.07^\circ$ for the largest model (64D). We use the joint rotations to compute SMPL

Figure 3.5: ***Visual Facial Animation Reconstruction.*** *Our Spline-based Transformer is able to successfully represent facial animations, preserving both the identity and expression of the subject throughout the performance.*

body model parameters [Loper et al., 2015] and visualize the reconstruction in Fig. 3.6 along with the reconstruction error in mm. Positional encoded transformers have recently received a lot of attention in full-body motion reconstruction and synthesis [Petrovich et al., 2021; Tevet et al., 2023; Duan et al., 2021]. We believe that Spline-based Transformers can help to increase the performance of various state-of-the-art models designed for these applications.

Motion Editing. We also show that motions can be modified by applying simple operations to the control points. In Fig. 3.6, we visualize two result-

Table 3.4: ***Human Motion Reconstruction.*** *Comparison across different latent dimensions.* ***Bold*** *indicates the best overall performance, and underline the best in each category. We report the Mean Squared Error between joint angles [deg].*

Method	16D	32D	64D
ALiBi	0.151	0.103	0.059
ALiBi-Cat	0.153	0.103	0.051
Spline (Ours)	0.054	0.022	**0.006**

Figure 3.6: ***Full Body Reconstruction and Modification.*** *Upper rows: Reconstruction quality on a motion. Lower rows: Two reconstruction results after modifying the control tokens.*

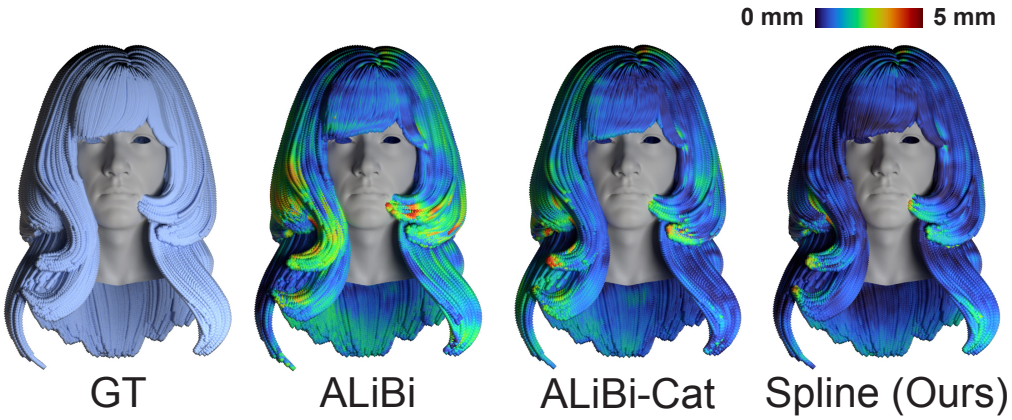
ing motions where the control points have been modified to be closer or further away from the end-points of the spline. The motions preserve the overall style but change in speed and detail. This experiment suggests that latent splines behave smoothly in a neighborhood and edits result in plausible motions. We observe that motions can easily be toned down or amplified as we show with more results in the accompanying video. The spline-based latent space further allows us to super-sample motions. By sampling the latent spline more densely before decoding, we can achieve up to a 4x upsampling of a motion clip. This method effectively preserves the original motion characteristics, as demonstrated in the video. Multiple splines can be combined to represent longer sequences of motions. Each of the segments can then be modified individually.

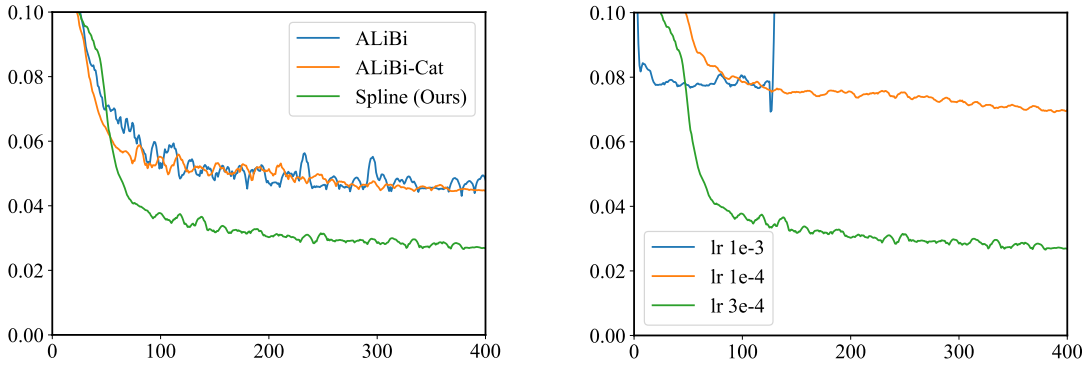
3.3.4 Geometric Representation

Finally, we show how Spline-based Transformers can also be used to model complex geometry like hair strands [Rosu et al., 2022; Zhou et al., 2023], which present themselves as 3D curves in space. For this experiment, we use a dataset of 343 unique 3D hairstyles [Hu et al., 2015], where each hairstyle contains 10000 strands, and each strand has 100 points. We are interested in representing only the strand geometry in our experiment, so we consider the strands across the hairstyles as individual 3D curves in space. We normalize the root position of each strand by translating it to the origin. Each normalized strand is therefore a sequence of a 100 vertices and is used to train a transformer autoencoder as before with an L2 reconstructive loss. We report the reconstruction error of strands from 10 test hairstyles in Table 3.5. While a thorough comparison to state-of-art methods [Rosu et al., 2022; Zhou et al., 2023] is required to demonstrate the real effectiveness of Spline-based Transformers for this task, our initial tests indicate that they could be an interesting architectural alternative. For the purpose of visualizing the reconstructed strands as a coherent hairstyle, we apply the ground truth root position to the reconstructed strands in Table 3.5.

Table 3.5: ***Strand Reconstruction.*** *Comparison across different latent dimensions.* ***Bold*** *indicates the best overall performance, and underline the best in each category. Performance is measured in Mean Squared Error (MSE).*

Method	8D	16D	32D
ALiBi	4.8e-3	2.0e-3	1.5e-3
ALiBi-Cat	4.6e-3	1.9e-3	1.2e-3
Spline (Ours)	1.09e-3	1.06e-3	**9.4e-4**





(a) *validation loss* **(b)** *learning rate sensitivity*

Figure 3.7: ***Training Performance (AFHQ).*** *(a) shows the validation loss of the different methods. (b) shows the sensitivity of Spline-based Transformers to the learning rate.*

3.3.5 Practical Limitations

Spline-based Transformers not only achieve a better performance than conventional transformers as demonstrated by our experiments, but can also achieve this performance improvement much faster than the traditional positional encoded models. Subfig. 3.7a shows the validation loss on an image experiment.

While they converge faster, we observe that the Spline-based Transformers are sensitive to learning rates. Subfig. 3.7b shows the same run with three different learning rates. A large learning rate can lead to a model collapse where the control points start to converge to the same point in latent space; leading to the same latent code for each token in the input sequence and the model not being able to recover from this state. We also observe that having a too small learning rate can harm performance significantly. We believe that a specialized scheduling strategy could significantly improve the stability and performance of Spline-based Transformers and leave this as future work.

3.3.6 Latent Space Visualization

After training, we can visualize the latent spline trajectories (see Fig. 3.8). The trajectories show similar characteristics to sinusoidals but are more complex and asymmetric curves. In some parts, the change of the features is more rapid, while other dimensions propagate the same value over the whole sequence length.

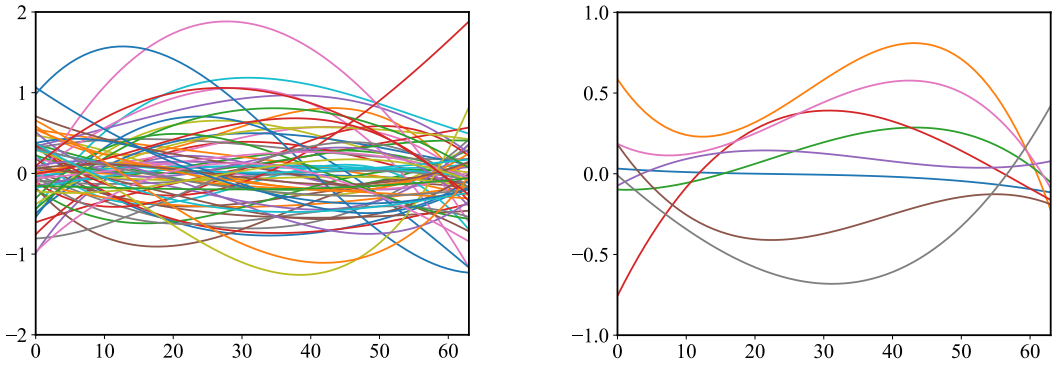


Figure 3.8: ***Latent Splines (AFHQ).*** *Visualization of predicted latent spline trajectories in $d = 64$ and $d = 8$. x-axis: position along the spline; y-axis: value of the feature.*

3.4 Concluding Remarks

Contributions. In this chapter, we introduced Spline-based Transformers, a new class of Transformer models that eliminate the need for absolute positional encoding by combining temporal and contextual information into a single trajectory, represented by a latent spline curve. We presented the superior performance of Spline-based Transformers across a variety of datasets, from simple curves to complex animation data and images. The experiments show significant performance improvements over traditional positional-encoded transformer models. Spline-based Transformers are trivial to implement, yet effective, and have no additional computational overhead. We identify improvements to the training stability as future work to reduce the sensitivity to training hyperparameters, such as the learning rate. The spline-based latent space introduced by our method opens up a new way to interact with latent spaces, enabling straightforward modifications of the latent control points. We hope that our work encourages research towards a new class of transformers with controllable latent spaces across a variety of applications.

Limitations. We identified training instabilities as an important direction for further research. For example, introducing additional regularization may help mitigate singularities in the latent space of Spline-based Transformers and reduce the sensitivity to training hyperparameters.

CHAPTER 4

Robust Motion Tracking

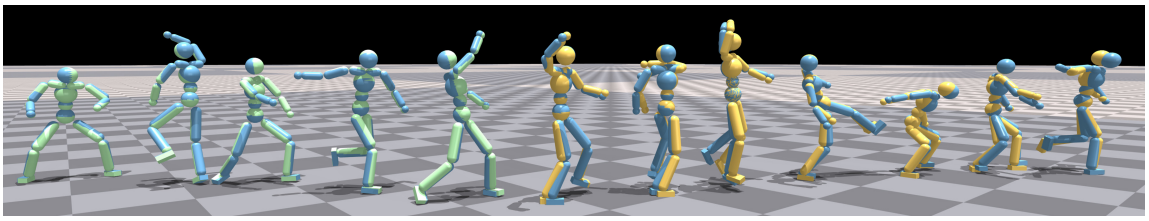


Figure 4.1: ***Robust Motion Tracking.*** *Our framework enables the control of a physics-based character using user-specified kinematic reference motion (in blue), mapping it to a sequence of temporally and spatially consistent latent codes (decoded in green). In a second step, we condition a reinforcement learning policy on the latent code, generating actuator commands that execute the motion (yellow) while obeying the laws of physics.*

Imitation-driven reinforcement learning has led to an astonishing leap of progress towards a long-term goal of physics-based character animation, namely accurate and robust tracking across diverse skills. Despite this progress, we lack techniques that achieve this goal with a single policy, covering the diversity in an unfiltered dataset of universal and dynamic motions while providing full-body control of the interactive character.

In this chapter, we propose a two-stage technique that takes kinematic ref-

This chapter is based on the publication *"VMP: Versatile Motion Priors for Robustly Tracking Motion on Physical Characters"*, Agon Serifi, Ruben Grandia, Espen Knoop, Markus Gross, and Moritz Bächer [Serifi et al., 2024b]. A supplemental video is available at `https://youtu.be/Q2I7u0tjlJs`.

erence motion as input and maps it to actuator commands of a character. In a first stage, we extract a latent representation of kinematic motion from an unstructured dataset of motion clips. To this end, we train a variational autoencoder to reconstruct kinematic motion, feeding it with short windows of motion data. In a second stage, we train a policy to imitate motions from the dataset, conditioned on both the current frame of the reference motion and the latent code that corresponds to a time-shifted window centered at the frame. After training, a user-provided kinematic reference is mapped to the latent space and then fed to the policy to control the character.

By training the latent space and control policy separately instead of end-to-end [Merel et al., 2018; Hasenclever et al., 2020; Won et al., 2022], we take advantage of well-studied self-supervised techniques to obtain a structured latent code that captures the diverse distribution of motions within a dataset. Conditioning on latent codes, we can train a single control policy with an explicit imitation reward that preserves the diversity in the input data, without the need for adversarial strategies [Peng et al., 2022; Dou et al., 2023; Tessler et al., 2023] that are prone to mode collapse, or a set of expert policies that are trained on clusters of similar motions [Won et al., 2020; Luo et al., 2023]. Taken together, the motion encoder and control policy enable the control of the full character for skills of high complexity, extracting motion features from short, overlapping subsequences of clips.

With a set of demonstrations on both virtual and a physical humanoid character, we show that a combination of known building blocks arranged in a novel way leads to a simple, yet effective technique that scales well when it comes to diversity and training complexity, tracks unseen dynamic motions closely and physically infeasible motions robustly, and interfaces with common animation techniques and character control modalities. Succinctly, we contribute

- A demonstration of enhanced downstream imitation learning through a pretrained versatile motion prior that provides a structured encoding of kinematic motion.
- A single control policy that is trained on an unfiltered, large-scale dataset of diverse human motions, providing accurate tracking and generalization to unseen input.
- A demonstration that our two-stage processing transfers to robotic characters in the real world, robustly executing expressive motions at the physical limits of hardware.

4.1 Related Work

Learning-based techniques for character control have received increasing attention in recent years, outperforming model-based techniques in the generation of life-like motions by interfacing with motion data. We focus our review on data-driven methods, in particular in the two broad categories of kinematic and physics-based motion synthesis.

Kinematic Motion Synthesis. In a learning-based approach to kinematic motion generation, a common goal is to create a compact representation of motion that allows for smooth temporal and spatial composition [Levine et al., 2012; Harvey et al., 2020; Starke et al., 2020; Starke et al., 2022], or to use auto-regressive models to learn the distribution of motion sequences [Ling et al., 2020; Rempe et al., 2021; Chandran et al., 2022a]. Recently, advanced generative models were used to synthesize kinematic motions [Tevet et al., 2022; Tevet et al., 2023; Shafir et al., 2023; Raab et al., 2023; Raab et al., 2024; Lee et al., 2023]. To make the generated motions physically plausible, geometric losses are used to reduce visual artifacts like foot sliding or penetrations. To fulfill stricter constraints, physics engines can be incorporated into the generation process [Won et al., 2022; Yuan et al., 2023].

These works use a latent space to generate output motions and are related to the first stage of our proposed processing. However, we focus on generating physically informed motion and use the embedding as an interface to a low-level controller of a physics-based character instead.

Physics-Based Methods. The challenges of designing general objectives that produce natural motions have motivated the use of data-driven techniques that imitate target animations [Sok et al., 2007; Wampler et al., 2014; Liu and Hodgins, 2017; Grandia et al., 2023]. Simple imitation objectives, together with advances in deep reinforcement learning (RL) [Sutton and Barto, 2018], yield high-quality physics-based character control [Peng et al., 2018a; Bergamin et al., 2019; Park et al., 2019b; Luo et al., 2020]. However, in these works, the policy is limited to the imitation of one or a handful of similar skills, requiring additional mechanisms to transition between more diverse skills.

To learn from large-scale motion datasets, sophisticated methods are proposed that balance and filter motions, or learn motion matching procedures to increase coverage [Bergamin et al., 2019; Wang et al., 2020]. Another way to make use of large heterogeneous datasets is to divide the data: Won *et*

al. [2020] propose a motion clustering followed by learning a mixture-of-experts, and Luo *et al*. [2023] propose to train a hierarchical policy where new policies are allocated for increasingly difficult motion sequences. While they can achieve impressive imitation quality on a diverse set of skills, we simplify the process by incorporating a self-supervised kinematic stage that enables the efficient training of a *single and simple* multilayer perceptron policy on a large corpus of data.

Another stream of work focuses on exploiting latent spaces that enable the reuse of policies in a more general setting, targeting a foundation model [Bommasani et al., 2021] for motion control. [Zhu et al., 2023; Merel et al., 2018; Hasenclever et al., 2020; Won et al., 2022; Gehring et al., 2023] jointly train a motion encoder with the policy to extract an embedding that can be reused in high-level RL tasks. Merel *et al*. [2020] extract a latent space by post-processing multiple expert policies, distilling their knowledge into a unified policy. To increase sample efficiency, a world model can be incorporated [Yao et al., 2022; Fussell et al., 2021; Feng et al., 2023], but does not scale to hours of data [Yao et al., 2022] or is sensitive to data quality [Fussell et al., 2021]. Additionally, adversarial learning has been used as an alternative to explicit imitation objectives [Peng et al., 2021; Peng et al., 2022]. However, this approach suffers from a long training time and is prone to mode collapse. A conditional discriminator mitigates mode collapse in these settings [Tessler et al., 2023; Dou et al., 2023], but does not prevent it. Moreover, due to the long training time, these methods are trained on relatively small datasets, which limits generalization to diverse, unseen motions. We show that a pre-trained latent space in combination with explicit imitation rewards results in a universal motion controller that avoids mode collapse and scales well to large datasets.

Directability of Characters. To direct the character, RL-based methods often use a combination of high-level policies, planners, or finite state machines that interface with low-level policies, bridging the gap between high-level commands or task specifications and actuator commands [Peng et al., 2021; Wang et al., 2020; Park et al., 2019b]. In this context, latent space embeddings were used as input to the RL policy [Peng et al., 2022; Merel et al., 2020]. However, in hierarchical approaches, explicit control of the resulting motion is lost. To give users more control, Juravsky *et al*. [2022] couple latent control with the latent space of a pre-trained language model [Radford et al., 2021]. Xu *et al*. [2023] present a method that enables spatial imitation of different motion clips while executing a high-level task. However, retraining is required for new tasks or styles.

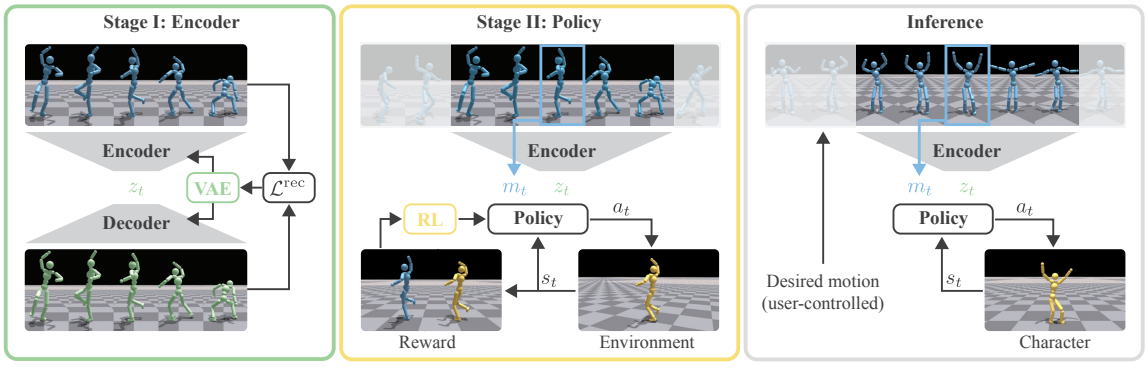


Figure 4.2: ***Overview.*** *Our two-stage processing starts by taking random samples of motion windows from a database of clips as input to train a variational autoencoder (VAE) to reconstruct them, extracting a latent kinematic motion space (Stage I). Feeding the encoder with time-shifted motion windows, we then train a policy, conditioned on the latent code of the window and the frame at its center, with rewards on the proximity of the simulated to the kinematic state at frame t (Stage II). The resulting encoder-policy pair provides full-body, motion control of the interactive physics-based or robotic character (Inference).*

Similar to previous work [Wang et al., 2020; Bergamin et al., 2019], we interface with common control modalities by providing fine-grained control of characters but generalize over a more diverse set of skills by utilizing a kinematic latent space. In summary, we are unaware of a technique that trains a single policy efficiently, provides the same level of coverage, robustness, and generalization properties on a large-scale dataset of complex motions, while also preserving high-fidelity full-body control of the character without the need for further training.

4.2 Two-Stage Processing

As outlined in Fig. 4.2, our processing separates the extraction of a kinematic latent space from the training of a dynamics-aware policy. In a first stage, we train an encoder-decoder pair to reconstruct short motion windows that we randomly sample from a large dataset of unfiltered motion clips. These clips are representative of the universal skills of a human, virtual character, or robot. In a second stage, we then train a control policy, conditioned on the kinematic state at the current frame and a latent code for a window around the frame, to maximize kinematic tracking and smoothness rewards. After training, we use the combination of the encoder and control policy to control the full-body motion of a character with unseen input motion. The user

interface is therefore the specification of a kinematic motion sequence, which allows for stylized and precise control of the character.

In the next two subsections, we will discuss the first two stages of our processing in detail, followed by an evaluation of our approach with an ablation study, analysis of coverage and smoothness of our latent space, and comparisons to related approaches in Sec. 4.3. In Sec. 4.3, we will also demonstrate applications of our approach in the fine-grained motion control of a physics-based and the control of a robotic character.

4.2.1 Extracting Latent Motion Priors

To temporally and spatially resolve skills at a fine-grained level, we extract the kinematic state, consisting of positions and velocities, for a few past and a few future frames, in addition to the center frame. We then feed these randomly sampled motion windows to a VAE architecture [Kingma and Welling, 2013], learning a latent space that captures the structure of the motion distribution and similarities between skills for a short time horizon around the current state. While velocities provide some information about what the future state of the character will be, velocities can suddenly change due to impact. Hence, it is crucial to be able to anticipate what the character will do in the near future, or know what has happened in the near past. The launch and landing for a jumping sequence are good examples of why context beyond the current state is important to capture in a latent space. However, to achieve generalization, it is important that the window is short enough, such that the latent space captures fundamental motion building blocks that will also appear in unseen motions and does not overfit to a particular training motion.

More formally, our first stage takes a dataset, $\mathcal{D}$, of distinct clips as input, consisting of a finite sequence of motion frames

$$m_t = \{h_t, \theta_t, v_t, q_t, \dot{q}_t, p_t\}. \tag{4.1}$$

h_t is the height of the character's root relative to the ground. While we ignore the absolute position of the root in the plane, the inclusion of the relative height is important to differentiate a jump where the height changes from walking on the ground where the height remains constant. θ represents the orientation of the root, expressed as a 6D vector [Zhou et al., 2019], and the 6D vector v_t its linear and angular velocity. q_t and $\dot{q}_t$ are the angular positions and angular velocities of all joints. In addition, we include the positions, p_t, of hands and feet, relative to the root, in the features for frame t.

To train our VAE, we extract motion windows of length $2W + 1$

$$M_t = \{m_{t-W}, \ldots, m_{t+W}\} \tag{4.2}$$

from individual clips. To normalize them, we first express the orientations, velocities, and end effector positions of individual frames in a local heading frame that we extract from the root pose of the center frame, making the normalized windows invariant to the heading direction. We then use the mean and standard deviation of quantities of all frames in $\mathcal{D}$ to normalize the quantities in the motion window frames, except for orientation, where we skip this second normalization.

The encoder of our VAE, $e_\psi(z_t|M_t)$, maps the motion window to a distribution of latents, $z_t \in \mathbb{R}^{d_z}$, and is modeled as a multivariate Gaussian distribution. The sampled latent representation is then mapped back to input space by a decoder, $M'_t = d_\phi(z_t)$. We train the β-VAE [Higgins et al., 2017] with a reconstruction loss on the motion window

$$\mathcal{L}^{\text{rec}}(M_t, M'_t) = \frac{1}{2W+1} \sum_{i=t-W}^{t+W} l^{\text{rec}}(m_i, m'_i), \tag{4.3}$$

and the weighted KL-divergence loss with a standard Gaussian distribution prior as the latent distribution. For individual frames, we compute the loss on standard normalized quantities, first computing rotation matrices for orientations using the Gram-Schmidt process

$$\begin{aligned} l^{\text{rec}}(m_i, m'_i) = \|h_i - h'_i\|_2^2 + \|R(\theta_i) - R(\theta'_i)\|_F^2 + \|v_i - v'_i\|_2^2 \\ + \|q_i - q'_i\|_2^2 + \|\dot{q}_i - \dot{q}'_i\|_2^2 + \|p_i - p'_i\|_2^2. \end{aligned} \tag{4.4}$$

Because we work with normalized quantities, no relative weighting is needed here.

After training, we prepare for the next stage by encoding motion windows for all frames of all clips, resulting in a latent code z_t per frame m_t. At the beginnings and ends of clips, we repeat the start and end frames to initialize complete windows. Note that in contrast to previous work [Peng et al., 2022; Juravsky et al., 2022; Tessler et al., 2023], we encode a clip with a sequence of latent codes instead of a single one, identifying similarities at a fine-grained level. Besides being advantageous for generalization, our embedding is crucial for precise control, because policies are less reactive otherwise and tend to finish the previous conditioned motion sequence before adapting to a new target.

4.2.2 Training Conditional Policy

In the second stage of our processing, we train a policy using a reinforcement learning framework [Sutton and Barto, 2018], where the agent interacts with the environment and maximizes the expected discounted return. At each time step, the agent produces an action a_t according to the stochastic policy, $\pi(a_t|s_t, c_t)$, where c_t is the conditional input to the policy, and s_t is the observed state at time t. Provided with the action, the environment then produces the next state, s_{t+1}, and a scalar reward $r_t = r(s_t, a_t, s_{t+1}, c_t)$.

We control our character with a policy that is conditioned on the time-varying latent code, z_t, and the instantaneous motion reference, m_t, *i.e.*, $c_t = (m_t, z_t)$. We normalize m_t with the same procedure as we use for motion windows, with $W = 0$. Our experiments show that adding both modalities is beneficial: The kinematic reference provides instantaneous feedback to the policy and improves tracking accuracy, while the latent code contains information about the intermediate past and future and helps the policy to bring the current target in alignment with similar motions.

During training, we initialize an episode of fixed length T by randomly choosing a frame from the dataset, retrieving the pair (m_t, z_t). We then shift by one frame within the same motion clip to retrieve the next pair. We continue this process until we reach the end of a clip, randomly sampling a new frame from a new clip if the episode has not terminated yet. The randomized initialization avoids the policy getting stuck within the starting sequence of motion clips and leads to an increased learning efficiency [Peng et al., 2018a].

Our reward contains a combination of motion tracking, staying alive, and regularization terms that mitigate high-frequency motion,

$$r_t = r_t^{\text{track}} + r_t^{\text{alive}} + r_t^{\text{smooth}}. \tag{4.5}$$

Following previous work on tracking-based imitation learning [Lee et al., 2010; Peng et al., 2018a; Park et al., 2019b], we compute rewards between the reference m_t and simulated pose of the character,

$$\begin{aligned} r_t^{\text{track}} = &- c^h \|h_t - \hat{h}_t\|_2^2 - c^\theta \|R(\theta_t) - R(\hat{\theta}_t)\|_F^2 - c^v \|v_t - \hat{v}_t\|_2^2 \\ &- c^q \|q_t - \hat{q}_t\|_2^2 - c^{\dot{q}} \|\dot{q}_t - \hat{\dot{q}}_t\|_2^2 - c^p \|p_t - \hat{p}_t\|_2^2, \end{aligned} \tag{4.6}$$

where quantities with a hat denote observations from the simulated state, normalized with the same procedure as we use for motion frames.

To allow ground contact with every body part, we apply early termination

on large deviations from the target state that persist for a longer time period, instead of the contact-based termination used in related work [Peng et al., 2018a]. Concretely, we terminate the episode if the maximum end-effector deviation, $\|p_t - \hat{p}_t\|_\infty$, exceeds a given threshold, t^p, for more than f frames.

A survival reward provides a simple objective that motivates the character to stay alive and prevents it from seeking early termination at the beginning of training,

$$r_t^{\text{alive}} = c^{\text{alive}}. \tag{4.7}$$

To mitigate vibrations and avoid unnecessary actions, we apply a first- and second-order action rate penalty, and penalize joint torques τ,

$$r_t^{\text{smooth}} = -c^{\Delta a}\|a_t - a_{t-1}\|_2^2 - c^{\Delta^2 a}\|a_t - 2a_{t-1} + a_{t-2}\|_2^2 - c^{\tau}\|\tau\|_2^2, \tag{4.8}$$

where the smoothness weights trade off tracking accuracy against the suppression of sliding or vibration artifacts.

Finally, we use domain randomization to increase the robustness of the policy and avoid overfitting to a single set of simulation parameters. The mass of each rigid body is randomized by a percentage error ϵ_m. We perform random pushes on the root, head, hands, and feet. Moreover, we randomize the frictional coefficient of the ground to prevent the policy from exploiting a particular coefficient through foot sliding or vibrations. To further reduce the sim-to-real gap, we added actuator models to the simulator. For robotic characters, we additionally perturb the joint positions by a maximum of ϵ_q to account for inaccuracy in joint calibration, and randomize the friction coefficient of the local ground plane to account for imperfections in the real world.

4.3 Evaluation and Results

Before we discuss evaluations and applications of our technique, we describe the large and diverse dataset (11 h) that we use for training, also detailing the network architectures and training procedures we use for the two stages.

4.3.1 Characters, Dataset, and Training Procedure

Characters. We evaluate our technique on a standard humanoid with 36 degrees of freedom (DOF) and a bipedal robot (Lima) with 20 DOFs. The

robot is 0.84 m tall and weighs 16.2 kg. Both characters are controlled with a set of torques, τ_t, which we compute from the actions with an actuator model [Grandia et al., 2024]: Taking the actions as inputs, we compute motor torques with proportional-derivative (PD) controllers

$$\tau_t^{\text{motor}} = \kappa_P \cdot (a_t - q_t) - \kappa_D \cdot \dot{q}_t, \tag{4.9}$$

where $\cdot$ denotes component-wise multiplication. We set the proportional and derivative gains, κ_P and κ_D, according to entries in Tab. 4.1 top. In addition, we compute frictional torques using a Coulomb model with a viscous component

$$\tau_t^{\text{friction}} = \mu_s \cdot \tanh(\dot{q}_t / \dot{q}_s) + \mu_d \cdot \dot{q}_t, \tag{4.10}$$

dividing (component-wise) joint velocities by static activation velocities, $\dot{q}_s$, and multiplying the two terms with the static and dynamic friction coefficients, μ_s and μ_d. We finally form the joint torques by clamping the motor torques and subtracting the frictional torques

$$\tau_t = \text{clamp}(\tau_t^{\text{motor}}) - \tau_t^{\text{friction}}. \tag{4.11}$$

To clamp the torques, we define velocity-dependent minimum and maximum torques. These limits consist of constant limit torques for braking and low velocities, τ_{max}, and linear limits ramping down the available torques above limit velocities, $\dot{q}_{\tau_{\text{max}}}$. The linear limits cross the maximum velocities, $\dot{q}_{\text{max}}$, when the limit torques are zero. For the two 5 DOF legs of the robot, we use Unitree-A1 (U-A1), and for its neck and arms Dynamixel-XH540-V150-R (D-XH540) actuators. We also use actuator limits for the humanoid, scaled to the size and expected dynamic performance of the character.

The observable state for the humanoid is a 217-dimensional vector consisting of the measured root height, root velocities, joint states, and key body positions, normalized with respect to the heading direction. Additionally, the actions of the previous two time steps are added to the state, which allows the policy to perform well on our smoothness rewards. For our bipedal robot, we omit the root height and key body positions from the state vector. The former is hard to accurately estimate in the real world, while the latter removes the need to compute forward kinematics on the robot. On the physical system, we use encoder measurements from the actuators, together with measurements from an on-board IMU, to estimate the robot's state [Hartley et al., 2020], incorporating motion capture data for increased accuracy.

Dataset. We use three sources of motion data: Reallusion (214 clips, 0.5 h) [2024], the CMU mocap dataset (1870 clips, 8.5 h) [2001], and Mixamo

Table 4.1: *Actuator, VAE, and RL Parameters.*

Actuator Parameters				
Param. entries	Units	Humanoid	U-A1	D-XH540
κ_P	$\mathrm{N\,m\,rad^{-1}}$	100	15	5
κ_D	$\mathrm{N\,m\,s\,rad^{-1}}$	1.0	0.6	0.2
$\dot{q}_s$	$\mathrm{rad\,s^{-1}}$	0.1	0.1	0.1
μ_s	N m	0.45	0.45	0.05
μ_d	$\mathrm{N\,m\,s\,rad^{-1}}$	0.023	0.023	0.009
$\tau_{\max}$	N m	500	34	4.8
$\dot{q}_{\tau_{\max}}$	$\mathrm{rad\,s^{-1}}$	20	7.4	0.2
$\dot{q}_{\max}$	$\mathrm{rad\,s^{-1}}$	100	20	7

VAE Parameters		VAE Training	
Param.	Humanoid & Robot	Param.	Value
β	0.002	Batch size	512
W	30	Number of epochs	50 000
d_z	64	Learning rate	0.003
Param.	0.8M	KL-scheduler cycles/ratio	7/0.5
		Warm restart T_0/T_{mult}	1000/5

RL Parameters			RL Training (PPO)	
Param.	Humanoid	Robot	Param.	Value
c^h	0.5	0.0	Batch size	8192×32
c^v	1.0	2.0	Mini-batch size	8192×8
c^θ	1.0	1.0	Clip range, e	0.2
c^q	1.0	7.0	Discount factor, γ	0.99
$c^{\dot{q}}$	0.1	0.1	GAE discount factor, λ	0.95
c^p	1.0	1.0	Number of epochs	5
c^{alive}	6.0	30.0	Desired KL-divergence	0.01
$c^{\Delta a}$	0.1	1.5	Max gradient norm	1.0
$c^{\Delta^2 a}$	0.01	0.45		
c^τ	$1 \cdot 10^{-5}$	$1 \cdot 10^{-4}$		
t^p	0.2 m	0.3 m		
f	3.0 s	3.0 s		
T	30.0 s	30.0 s		
ϵ_m	10.0 %	10.0 %		
ϵ_q	0.2 rad	0.2 rad		

(2150 clips, 2.0 h) [2024]. They span highly diverse motions, from simple walking cycles to acrobatic motions that only very skilled humans can perform. While the Reallusion and Mixamo datasets are artist-processed and hence clean, the CMU dataset contains infeasible motions and noisy data. We work with the full and unfiltered dataset. To retarget the kinematic motions onto our robotic character, we use an inverse kinematics formulation [Schumacher et al., 2021].

VAE. The variational autoencoder [Kingma and Welling, 2013] for our motion latent space is built with 1D-convolutional layers, followed by 4 convResnet blocks without bias components [Dhariwal et al., 2020], Layer Normalization [Ba et al., 2016], ReLU activations [Fukushima, 1969], and a final linear layer. We use a latent code dimension of $d_z = 64$ and a window size of $W = 30$, amounting to 1 s. At the bottleneck layer, we double the encoder output dimension and sample from a multivariate Gaussian distribution. The decoder mirrors the encoder with deconv-layers.

For each training iteration, we extract a batch of 512 randomly sampled windows. We use the training objective of a β-VAE [Higgins et al., 2017] with a KL weight of 0.002 and cyclical scheduling to mitigate KL vanishing [Fu et al., 2019]. We use the RAdam optimizer [Liu et al., 2020] with an initial learning rate of 0.003, adapted using a cosine annealing scheduler with warm restarts [Loshchilov and Hutter, 2017]. The VAE is trained on an RTX 4090 for 10 h. See Tab. 4.1 middle for hyperparameters.

RL. We keep our policy network small and model it as a diagonal multivariate Gaussian distribution with the mean given by a neural network with ELU activations [Clevert et al., 2016] and 3 hidden multilayer perceptron (MLP) layers of 512 units. We simulate the character using the GPU-accelerated simulator in Isaac Gym [Makoviychuk et al., 2021], with 8192 environments simulated in parallel on an RTX 4090.

For training, we use Proximal Policy Optimization (PPO) [Schulman et al., 2017] and train for 48 h. To represent the value function we use the same architecture as for the policy. Observations are normalized using a running mean. See Tab. 4.1 bottom for reward and PPO parameters.

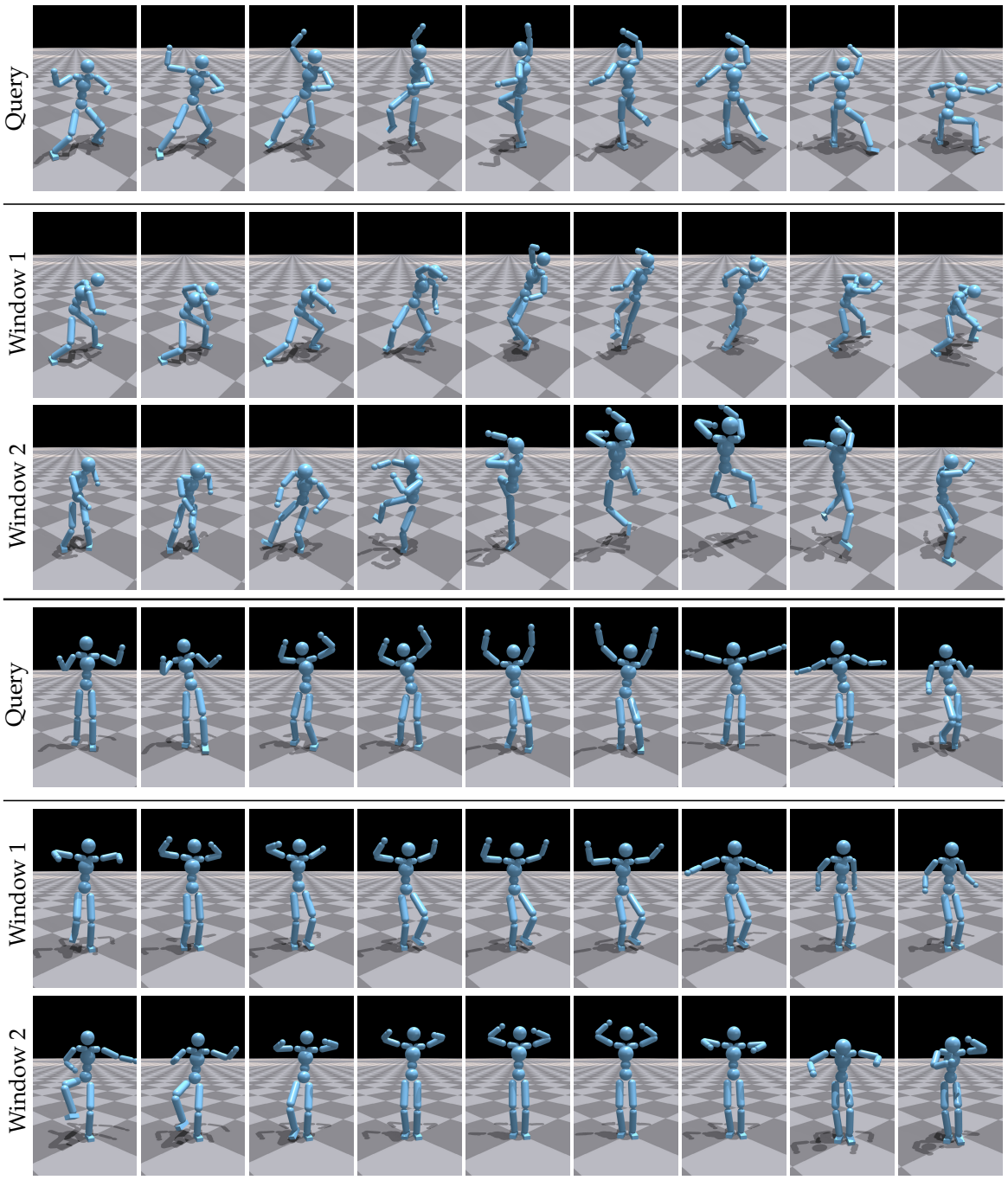


Figure 4.3: ***Latent Similarity.*** *Given a query motion window from the dataset (top) or unseen input (bottom), we compute the cosine similarity with all other windows using their latent representation. For each query, we show frames from the window with the highest (1) and second-highest (2) similarity score.*

4.3.2 Evaluation of Motion Embedding

The goal of our kinematic latent embedding is to capture similarities between short-horizon motion windows around individual frames. To enable generalization, motion windows from unseen kinematic input should map to latent codes that are in proximity to those of similar motion windows within the dataset. To validate that our latent space has desirable properties in this regard, we evaluate the latent space for the standard humanoid and show in Fig. 4.3 a randomly chosen window from our dataset (top query) and compare it to the two motion windows within the dataset that are closest in latent space. We repeat this experiment for an unseen user-specified input motion (bottom query).

In the video accompanying this chapter, we also evaluate the smoothness of our latent space by linearly interpolating between the latent space trajectories corresponding to two different clips. After interpolation, we decode this sequence and visualize the middle frame of the decoded window as a new motion clip. This results in natural in-between skills and demonstrates that our latent space provides the expected smoothness. We also visualize the reconstruction of unseen motion windows, exhibiting the expected coverage and generalization.

Comparison with End-to-End Latent Codes. CALM [Tessler et al., 2023] trains a latent representation end-to-end with the control policy. We compare the quality of the resulting representation by assessing the separability between motion classes within the latent space using Linear Discriminant Analysis (LDA) [Bishop, 1995] and by quantifying the mutual information between input motion and latent codes. To facilitate comparison, we train the first stage of our method on their dataset and use the same 2 s window length. Upon encoding the entire dataset using both approaches, motion classes for LDA are constructed by labeling consecutive 2 s of latent codes as a single class, with the subsequent 2 s being omitted to ensure distinct classes.

For mutual information analysis, the same dataset is utilized, and the latent codes are discretized into 100 bins per dimension. Mutual information scores are then computed for each pair of features. The results of LDA accuracy and mutual information in Tab. 4.2 demonstrate that our latent space preserves more information and exhibits superior separability of motions compared to the end-to-end method.

Table 4.2: ***Latent Space Comparison.*** *Comparison between end-to-end latent codes from CALM and the latent codes produced with our pretrained encoder. The pretrained latent space achieves higher Linear Discriminant Analysis accuracy (LDA acc.) and higher Mutual Information scores (MI) between state and latent code. This indicates that the latent codes preserve higher information quality than what the end-to-end encoder is able to extract.*

Method	LDA acc. ↑	MI ↑
CALM	0.687	0.121
Ours	**0.854**	**0.341**

Table 4.3: ***Ablation Study - Pose.*** *Motion Tracking MAE [joints \| end-effectors] [deg \| m].* ***Best*** *overall and best without latent codes. The RL policy inputs are based on motion input only (M), latent code (L) or both (LM).*

Input	Motions: Idle		Walk		Attack		Dance		Unseen	
M	7.52	0.037	8.63	0.044	13.11	0.065	12.79	0.057	13.29	0.075
M5	6.87	0.042	8.69	0.047	12.52	0.064	13.47	0.060	13.33	0.077
M10	7.96	0.038	9.06	0.044	12.93	0.062	13.76	0.058	13.60	0.079
L	6.10	0.040	7.53	0.054	10.70	0.069	10.45	0.064	10.92	0.072
LM	**4.31**	**0.031**	**4.15**	**0.037**	**7.08**	**0.060**	**5.80**	**0.046**	**7.83**	**0.069**

4.3.3 Evaluation of Two-Stage Processing

We first ablate our proposed choice of conditional policy inputs using a smaller dataset. Subsequently, we discuss the enhanced tracking performance and generalization potential achieved through training on a larger dataset. We then investigate the robustness of our policy in the face of infeasible inputs, disturbances, as well as its responsiveness to input discontinuities. All evaluations are performed with the standard humanoid.

Ablation. For our ablation study, we train the latent space as well as variants of our policy on the Reallusion dataset (0.5 h) and evaluate tracking performance qualitatively (Fig. 4.4, accompanying video) and quantitatively (Tab. 4.3). For evaluation of a policy, we generate 1024 fixed-size episodes with the same procedure as we use for training (Sec. 4.2.2), restricting the random sampling of clips to categories of increasing difficulty from Reallusion (Idle, Walk, Attack, Dance) and a small, randomly chosen subset of motions from the other two datasets (Unseen). To compare tracking perfor-

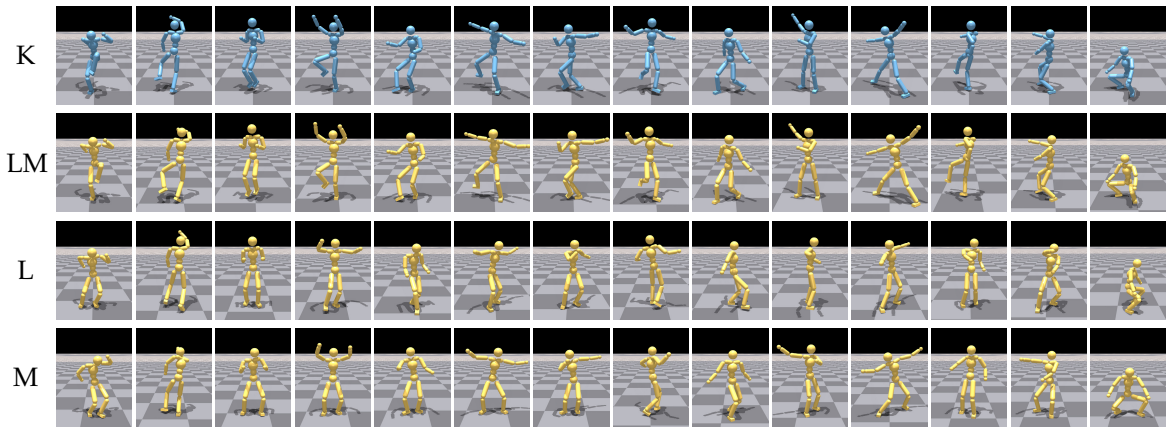


Figure 4.4: ***Tracking Performance.*** *Our proposed method (LM) tracks a kinematic input (K) significantly better than policy variants that are conditioned on the latent code (L) or the reference motion (M) only.*

mance, we report the mean absolute error (MAE) of joint positions over all frames, joints, and evaluation episodes (Tab. 4.3).

We first condition the policy on a single motion frame (M) and on motion windows with 5 (M5) and 10 (M10) additional future frames. These variants are similar to related approaches that target the fine-grained control of characters (*e.g.*, DReCon [Bergamin et al., 2019], UniCon [Wang et al., 2020]), omitting the use of additional mechanisms or policies for a fair comparison. As expected, the tracking error is higher for more challenging motions or unseen data. While the performance is good for less complex motions, it degrades quickly for more challenging or unseen input, visually perceived as the result of a low-pass filter. Similar to UniCon [Wang et al., 2020], we did not observe significant benefits from additional future frames.

In the lower half of Tab. 4.3, we report numbers for a variant that is only conditioned on the latent code (L), together with our proposed processing (LM). While the L-variant already consistently improves tracking performance, our LM variant clearly outperforms all other variants in all categories, especially for highly dynamic or unseen motions (Fig. 4.4; $\sim$50 % reduction in tracking error for Dance).

Comparison with End-to-End Method. To visually compare our method to CALM [Tessler et al., 2023], we sample random motion windows from the dataset. For CALM, we compute the corresponding latent code and condition the policy with the same code for a 2 second window, aligning with their training strategy. We then track the same motion using our pipeline.

Table 4.4: ***Scaling Capabilities.*** *We evaluate the scaling capabilities of each component when trained on more data. As the baseline, we train the VAE and RL on a smaller portion of our dataset and report tracking accuracy on (Unseen) data (VAE Small - RL Small). When increasing the data used for VAE training alone (VAE Full - RL Small), we already observe a positive effect, suggesting that better latent codes are helpful during RL training. Using more data for both stages further improves the tracking accuracy (VAE Full - RL Full).*

	RL Small	RL Full
VAE Small	7.83	-
VAE Full	**6.62**	**5.03**

The video presents the results side-by-side. Note that CALM requires two weeks of training on an industrial-grade A100 GPU, whereas our complete method trains in under three days on a consumer-grade RTX 4090. Retraining CALM on a consumer-grade GPU would take months. As demonstrated in the video resource, our method matches CALM's motion quality while exhibiting fewer artifacts. Our stronger coupling between latent code and target motion allows for easier control, whereas CALM's latent-only conditioning can result in repeated motion and is less reactive.

Generalization. To achieve generalization, it is essential to train on a large dataset. In our two-stage processing, the additional conditioning on the latent code during RL has a negligible effect on iteration time compared to the simple M-variants. This enables us to scale our approach and train a policy on a significantly larger dataset compared to related end-to-end approaches (ASE [Peng et al., 2022], PADL [Juravsky et al., 2022], CALM [Tessler et al., 2023], CASE [Dou et al., 2023]). To demonstrate the utility of our pre-trained latent space, we use the encoder and policy trained on the small Reallusion dataset as a baseline. Next, we use the small dataset during RL while using an encoder that is trained on the full set, with the exception of clips that we use for evaluation (Unseen). This already reduces the MAE tracking error on unseen motions by 15 % (see Fig. 4.4), indicating that the generalization performance of the policy is enhanced by improving the encoder.

By training both the encoder and the policy on the large set (except Unseen), we reduce the error to about 5° on unseen data. In the accompanying video, we provide a visual comparison of these three variants on a sequence of unseen dancing and fighting motions. We observe excellent tracking performance with our single policy that is trained on the full set, with noticeable artifacts for the variant that uses only the Reallusion dataset for both stages.

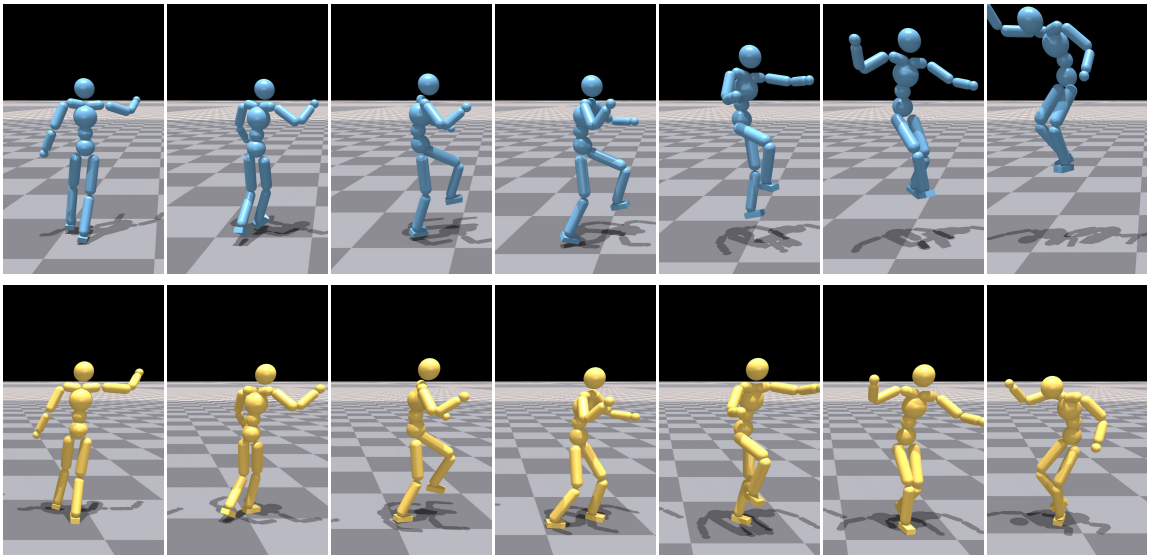


Figure 4.5: ***Robustness.*** *Conditioned on frames from an infeasible reference motion, the character remains stable and tries to track the performance as well as possible.*

Robustness. Our policy is robust to motions that are far from feasible, as shown in Fig. 4.5 and demonstrated in our video: The policy tracks the in-air stair climbing sequence as closely as possible while maintaining dynamic balance on the ground. In our video, we also demonstrate robustness under heavy disturbances by applying random forces and torques to the root, head, hands, and feet of the character. Additionally, on previously unseen uneven terrain, the character makes necessary adjustments to maintain balance and tracks the intended motion whenever possible. The method reaches its limits when the target motion is fast and physically unattainable. In such cases, the policy might either not track the motion or result in failure, as demonstrated in our video.

Reactivity. We evaluate the capability of the policy to abruptly transition between motions in the accompanying video. Since the policy is conditioned on a stream of latent codes rather than a single code for a longer sequence, it reacts instantly. This means that a user can arbitrarily sequence motions, without the need for smooth transitions. The policy is able to quickly adapt even if there are discontinuities in the input stream.

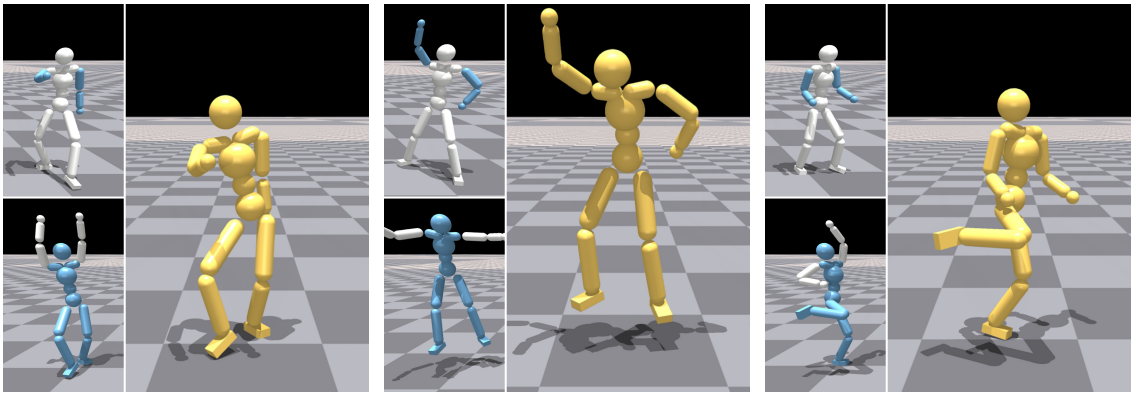


Figure 4.6: ***Spatial Composition.*** *Input motions can be spatially composed (selected bodies visualized in blue) without any additional training. Our method tracks the new motions as closely as possible within physical limits.*

4.3.4 Directability

Our policy interfaces with kinematic reference motion, generalizes well to new motions, and is robust to irregular input. It, therefore, seamlessly integrates into the standard workflows of artists and allows them to directly control a physics-based character without additional training. Hereafter, we present examples of how an artist might use our technique.

Spatial Composition. Our policy allows users to control the character by spatially composing motions of different body parts as illustrated in Fig. 4.6 and our video with three examples where arm motions are sourced from a different clip than the body motion. Even for random spatial compositions of motions that can result in implausible or unnatural input, we observe that our policy tracks the motion as closely as possible within the physical limits.

Motion Editing. With our technique, users can sequence full or partial motion clips in an arbitrary order to quickly create an initial reference animation for a character, as illustrated in Fig. 4.7 for a fighting sequence (Start). Because our method provides full-body control, they can then edit the motion reference to precisely time key events at key locations to achieve a particular task (Precision, hitting a pillar), followed by a stylization pass (Stylization).

Artist-Created Motion. We additionally asked an artist to create a dancing sequence with leg, arm, and full-body movements of increasing complexity with standard animation tools and workflows that are not physics-aware. As

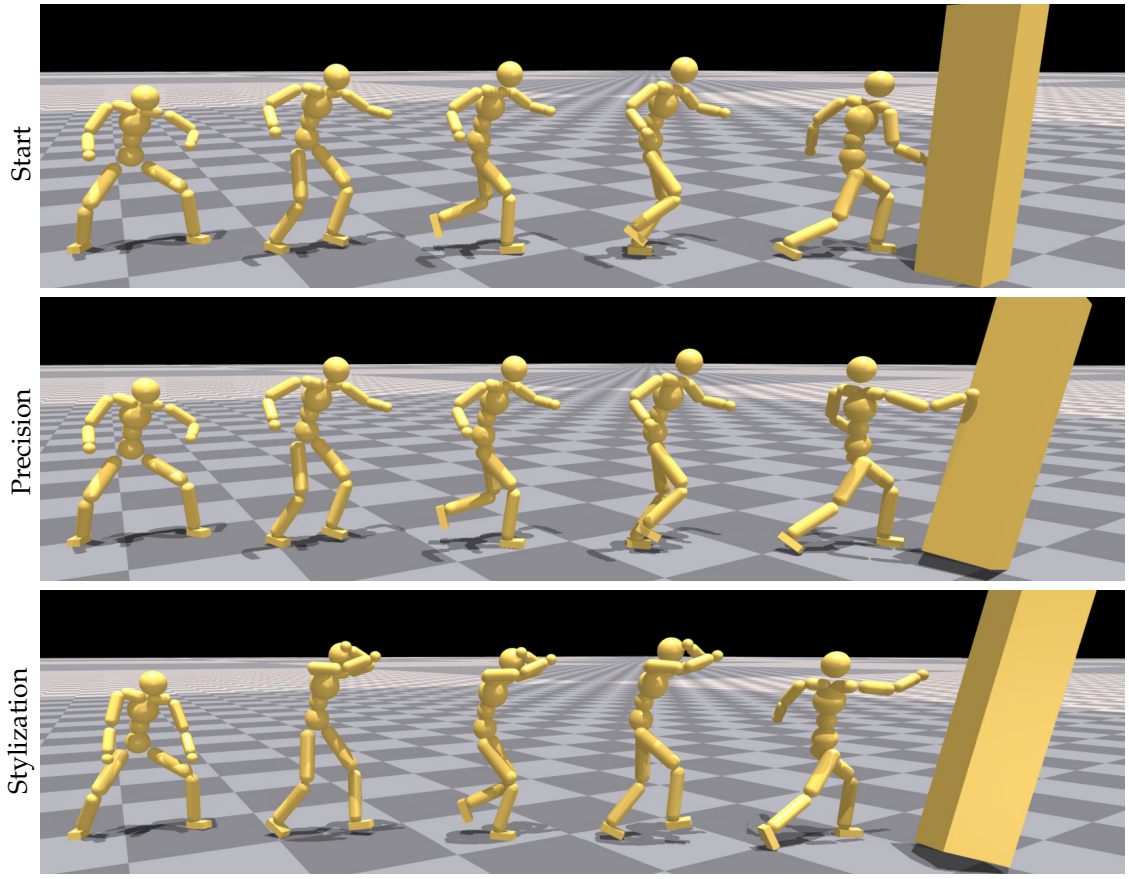


Figure 4.7: ***Motion editing.*** *Provided with an initial motion sequence (Start), a user can adapt the reference motion to precisely control the character (Precision) or change the style (Stylization).*

is common for artist input, there is visible foot sliding. Our method robustly tracks the performance and removes all foot sliding (see video). Because our inference is interactive, an artist could use our technique during animation to create physics-informed motions.

4.3.5 Robot Control

Recent progress in robotics in bridging the so-called sim-to-real gap [Zhao et al., 2020] provides evidence that extended domain randomization [Peng et al., 2018b] and the inclusion of actuator dynamics in the simulation [Hwangbo et al., 2019] can facilitate real-world deployment at the expense of presenting the RL algorithm with a significantly harsher training environment. We show that our presented pipeline, with actuator and learning parameters as summarized in Tab. 4.1, and domain randomization

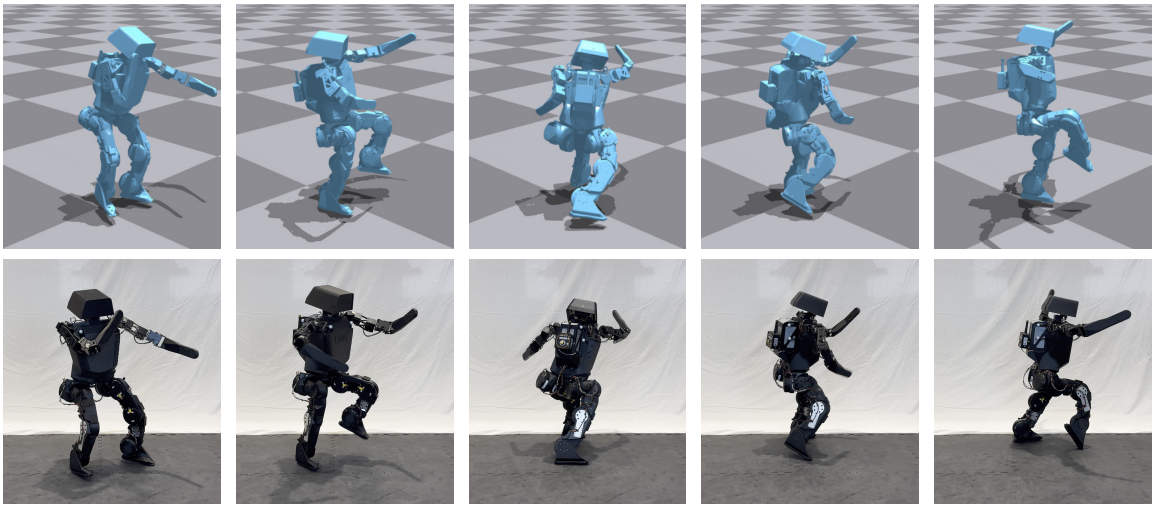


Figure 4.8: ***Lima Robot.*** *Transfer of dynamic motion skills onto a bipedal robot. The policy tracks the style as accurately as possible while maintaining balance.*

as discussed in Sec. 4.2.2, leads to a robust and effective policy that allows us to transfer versatile motions to a bipedal robot. Fig. 4.8 shows an example of a reference motion (top) and the resulting execution on hardware (bottom). Because our robotic character does not have an ankle-roll actuator, it cannot balance on a single leg for an extended amount of time. Interestingly, as shown in the last frame, the policy adapts by placing the tip of the right leg on the floor such that the reference pose can be closely matched without losing balance. The video shows additional demonstrations of highly dynamic motions. The quality of motion tracking remains high, even when constrained by the practical limits of physical actuators. Our results demonstrate that our method is effective on real robotic characters, showing that we do not rely on artificially strong simulated characters, external helper forces, or simulation artifacts.

4.4 Concluding Remarks

Contributions. We have shown that a pre-trained latent representation of motion improves both the tracking and generalization performance of a downstream RL-based control policy. Our two-stage training allows us to robustly track a diverse set of skills. Moreover, our kinematic motion interface empowers users to craft character-specific animations and have full-body control over physics-based characters. While not demonstrated explicitly, our approach would also interface with other common control modalities. Recent kinematic motion generators [Tevet et al., 2023;

Guo et al., 2022] could be used together with the policy to solve generative tasks in a physical environment.

Limitations. However, while our method has proven its strength on a diverse set of motions, it has difficulties in imitating clips that require a longer planning horizon. Acrobatic motions like backflips require a certain commitment to the performance. We believe that simple MLP policies cannot achieve generalization to this class of motions, requiring more sophisticated architectures with hidden states to accomplish coverage over unseen acrobatic input with extended flight phases. Moreover, while our method can track a kinematic reference, we have yet to explore generative capabilities of our processing, requiring an additional mechanism to traverse our time-varying latent space.

By demonstrating expressive motions on robotic hardware, we have unified recent progress in both the computer graphics and robotics communities. We see great potential at the intersection of these two fields and are excited about the future avenues of bringing increased agility and expressivity to more physical characters. In particular, we believe that the combination of self-supervised and reinforcement learning on large datasets provides a path to universal control policies that serve as a foundation model for a wide range of downstream tasks.

CHAPTER 5

Robot Motion Generation

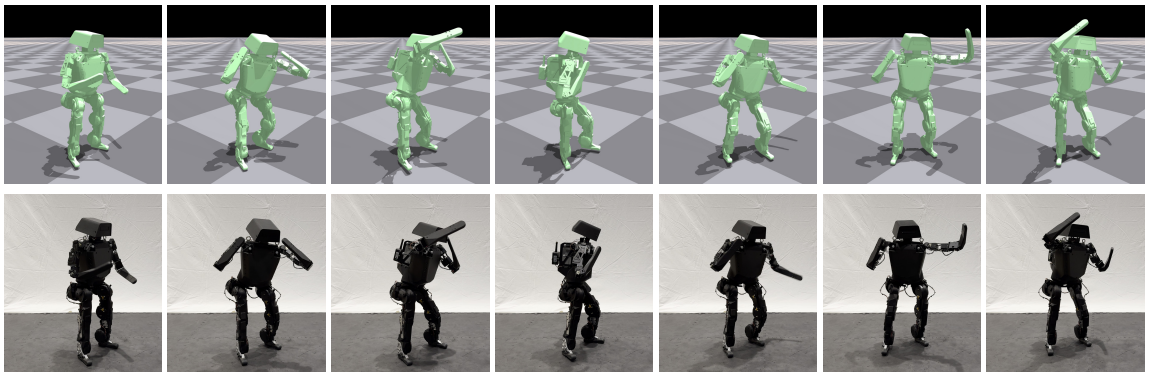


Figure 5.1: ***Robot Motion Diffusion Model.*** *RobotMDM generates motions that are physics-aware and respect character limits. Our method enables the seamless integration of kinematic motion generators with physics-based character control and can be deployed on robots. The example shows a robot performing the prompt "a person who performed a right-handed uppercut."*

The automated generation of realistic motions based on high-level user input is a highly relevant task in physics-based character animation and robotics. Traditionally, computer animation has emphasized kinematic-based approaches, which are well-suited for animated film and video games

This chapter is based on the publication *"Robot Motion Diffusion Model: Motion Generation for Robotic Characters"*, Agon Serifi, Ruben Grandia, Espen Knoop, Markus Gross, and Moritz Bächer [Serifi et al., 2024a]. A supplemental video is available at `https://youtu.be/eRXS98c_Suc`.

where visual storytelling takes precedence. Recent advances in generative models have demonstrated the ability to synthesize diverse and visually appealing motions when trained on large datasets [Tevet et al., 2023]. However, these kinematic-based generated motions do not strictly satisfy the constraints of a physics-based environment. As a result, the motions often contain artifacts such as floating, foot sliding, self-collisions, violations of joint limits, and dynamic imbalance, making it challenging to deploy these models in the real world. Although robust motion tracking controllers exist [Peng et al., 2018a; Wang et al., 2020], the resulting motion is inherently limited by the quality of the provided target motion. We therefore identify the need to *align* the output of kinematic generative models with the downstream task of tracking these motions with a physics-based or robotic character.

Evaluating the performance of a controlled character on generated motions requires long-horizon simulations, which are computationally expensive and non-differentiable. Even if a differentiable simulation is available, the highly non-linear nature of the articulated rigid body system and the contact dynamics results in poorly behaved gradients [Hämäläinen et al., 2020; Suh et al., 2022]. Drawing inspiration from Reinforcement Learning from Human Feedback (RLHF) [Christiano et al., 2017], we propose to train a reward surrogate that predicts the expected performance of the downstream task. This provides a differentiable and computationally efficient loss function to fine-tune the generative model. During deployment, we interface the fine-tuned generative kinematic model with the existing tracking controller.

This processing contrasts the direct training of a generative controller [Juravsky et al., 2022], which typically results in a controller with a latent space that can be sampled. However, since these controllers are trained with Reinforcement Learning (RL), they are typically constrained to shallow Multilayer Perceptrons (MLPs) and do not scale well to large datasets. By decoupling the problem of generating motion from tracking motion, we can utilize more advanced networks and specialized training strategies. This approach, which combines strong kinematic motion generators with imitation-driven physics-based controllers, directly scales to larger datasets. In this chapter, we build on a text-conditioned diffusion-based approach [Tevet et al., 2023], although our fine-tuning strategy is applicable to generative models in general.

Succinctly, our contributions include:

- A fine-tuning method for generative kinematic motion models that uses a reward surrogate, offering a computationally efficient, differentiable estimate of the downstream task.

- A demonstration of RobotMDM, a text-conditioned kinematic diffusion model that interfaces with an RL-based tracking controller, deployed on a real-world robot.

5.1 Related Work

Kinematic Motion Synthesis. Motion generation has been a pivotal area of research within computer graphics, primarily focusing on synthesizing realistic and context-aware motion for animated characters. The underlying goal of motion synthesis is to learn a controllable latent manifold from which natural motions can be drawn. In recent years, many neural architectures and different motion representations have been investigated [Holden et al., 2015; Holden et al., 2016; Holden et al., 2017; Lee et al., 2018; Harvey et al., 2020; Starke et al., 2019; Starke et al., 2020; Starke et al., 2022; Ling et al., 2020; Rempe et al., 2021; Chandran et al., 2022a]. This branch of work has been primarily used in animated character control, where the user provides simple control signals such as walking direction and velocity.

With the increased availability of unified large-scale motion capture datasets, such as AMASS [Mahmood et al., 2019], and comprehensive text and action annotations [Guo et al., 2022; Guo et al., 2020; Plappert et al., 2016; Punnakkal et al., 2021], progress has been made in generating expressive and diverse motions from more complex control signals. Guo *et al*. [2022] use an autoencoder combined with a recurrent neural network and text embeddings to generate motion sequences. In transformer-based extensions, a motion encoder and a text encoder are either trained jointly [Petrovich et al., 2022], or the motion latent space is aligned with a pre-trained language-image model such as CLIP [Radford et al., 2021]. This alignment exploits the rich semantic space of languages, enabling even the translation of cultural references into motions [Tevet et al., 2022]. More recently, T2M-GPT [Zhang et al., 2023] and MotionGPT [Jiang et al., 2024] formulate text-to-motion as a translation problem.

Diffusion models [Ho et al., 2020] have also been successfully adapted to the motion domain [Tevet et al., 2023; Zhang et al., 2024; Chen et al., 2023]. Beyond producing remarkably high-quality motion, these models inherit several key properties of diffusion models, such as supporting many-to-many generation and motion editing. Much research has leveraged Motion Diffusion Models (MDM [Tevet et al., 2023]) as foundational frameworks for motion synthesis. PriorMDM [Shafir et al., 2023] fine-tunes MDM to control the position of end effectors. Similarly, GMD [Karunratanakul et al., 2023] pre-

dicts a trajectory based on the given text prompt, which then guides the diffusion process. OmniControl [Xie et al., 2024] enables dense spatial control over any joints of the character. Extending this framework, DNO [Karunratanakul et al., 2024] introduces an optimization process where a differentiable objective is defined and used to optimize the input noise so that the resulting motion minimizes the objective. Although highly effective, this method depends on the differentiability of the objective. PhysDiff [Yuan et al., 2023] utilizes a pre-trained MDM and projects the motion onto a physically plausible state using a tracking controller and simulation. However, the evaluation of multiple simulations at runtime makes the method computationally expensive. Furthermore, the used control policy can apply non-physical residual forces to the character to preserve the semantic meaning of complex motions, aiming to remove visual artifacts such as ground penetration and floating. In contrast, we aim at incorporating physical understanding directly into the sampling process of MDM, and generate motions that are feasible without artificial external forces.

Physics-Based Character Control. Motion trackers, as discussed in Chapter 4, can follow very dynamic and complex motion references. And we presented a method that scales to large datasets and, beyond that, can be executed in the real world on a humanoid robot.

Besides imitation, the field has also studied latent spaces for generative tasks. One goal is to reuse imitation policies in high-level tasks, where an additional policy learns to navigate the latent space so that the generated motion reaches a goal. This latent space is either learned jointly with the imitation objective [Peng et al., 2022; Dou et al., 2023; Tessler et al., 2023; Gehring et al., 2023; Yao et al., 2022; Feng et al., 2023; Won et al., 2022; Zhu et al., 2023], or is exploited in a post-processing step [Merel et al., 2018; Luo et al., 2024]. To generate motions from text, Juravsky *et al*. [2022] propose to align the motion latent space with the latent space of a text encoder, similar to the kinematic counterparts [Tevet et al., 2022]. Albeit promising, the combined learning of policy and text conditioning suffers from the sample inefficiency of reinforcement learning, which restricts scalability to datasets of a few minutes and limits versatility. More recently, high-level neural networks based on transformers [Yao et al., 2024] and diffusion models [Ren et al., 2023] have been trained to navigate the pre-trained latent space of low-level policies. Despite their ability to model complex language-to-motion relations, current methods lack an understanding and notion of feasibility, resulting in invalid states or unnatural transitions.

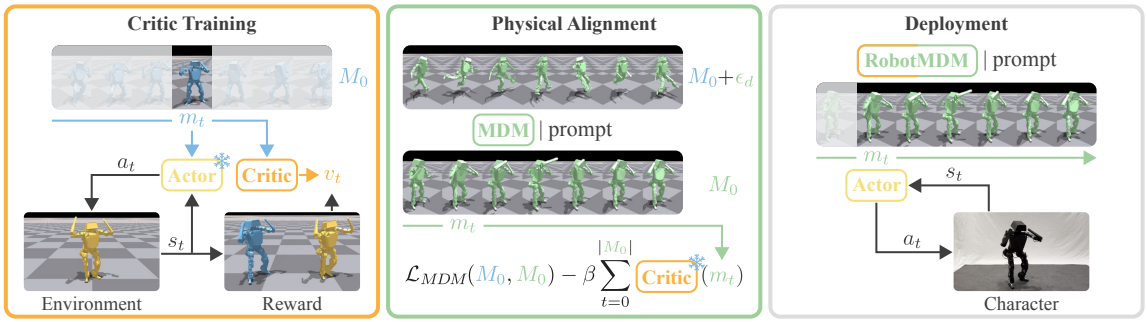


Figure 5.2: ***Overview.*** *RobotMDM leverages a pre-trained imitation policy (Actor) and a pre-trained Motion Diffusion Model (MDM) in a two-stage process. In the first stage (Critic Training), a Critic is trained using motions from the dataset to evaluate the Actor's performance, creating a differentiable surrogate for expected future rewards conditioned on motion input, and linking kinematic inputs to physical feasibility. In the second stage (Physical Alignment), the learned Critic is used to fine-tune the MDM, aligning it with the character's limits and ensuring physical feasibility. The final result is RobotMDM, a method capable of generating physics-aware motions, suitable for deployment on real-world systems (Deployment).*

Marginalized Critics. Marginalized critics, which predict the expected return of an RL agent based solely on context rather than both context and current state, have been utilized to shape a training curriculum that oversamples underperforming scenarios [Won and Lee, 2019; Xie et al., 2020]. In this work, we demonstrate that this critic formulation can also be applied outside the context of RL, serving as a loss function in a second learning problem. By leveraging the critic as a surrogate for the physical feasibility of generated motions, our method yields motion generators that better align with the physical character and control requirements.

5.2 Method

We assume the availability of a control policy, conditioned on a reference motion, and a generative model that produces kinematic motions. In this work, we train the VMP policy proposed in the previous chapter and an MDM generative model [Tevet et al., 2023] on our dataset. From there, our method consists of three parts (see Fig. 5.2): training a reward surrogate for the motion tracking task, aligning a generative model with this reward, and sequencing the generative model with the tracking controller during deployment.

Motion Representation. Motions of duration n are encoded with a $n \times (7 + 2j)$ matrix M, where j presents the number of joints. This matrix includes measurements for root height, root linear velocity (xy-plane), root angular velocity (about z-axis), root pose (3-dimensional), and joint positions and velocities. This representation is consistently applied across all stages of the method. Furthermore, motion data is normalized to the local heading frame of the character, where the x-axis aligns with the heading direction and the z-axis points upward. This normalization strategy decouples each frame from its absolute position and orientation in global coordinates, thereby facilitating a more efficient utilization of the data resources. Matrix m_t is a subset of rows from matrix M corresponding to either a single frame or motion window. If a motion is shorter, we pad the matrix with zero columns, restricting evaluations of loss or reward functions to the number of non-zero columns.

5.2.1 Critic Training

Conditional Reinforcement Learning. The goal in physics-based motion tracking is to translate kinematic motion inputs to physical actions in an environment. Using reinforcement learning [Sutton and Barto, 2018], a policy is trained through interaction with a simulated environment, maximizing the expected return over a period of time. The policy is a probability function, $\pi(a_t|s_t, m_t)$, where a_t is the action taken, s_t is the observed state at time t, and m_t represents the kinematic motion input to the policy[1]. The environment reacts to the action by transitioning to the next state, s_{t+1}, and providing a scalar reward $r_t = r(s_t, a_t, s_{t+1}, m_t)$. The reward reflects how accurately the resulting physical motion tracks the kinematic input. See Chapter 4 for a detailed specification of the reward and termination conditions.

During training, we initialize an episode by randomly choosing a motion and a starting frame from the dataset. We then shift by one frame within the same motion clip to retrieve the next reference. We continue this process until we reach the end of a clip, randomly jumping to a new clip if the episode has not terminated yet. Additionally, we employ domain randomization to enhance the robustness of the policy by randomly varying rigid body masses and friction coefficients, thereby avoiding overfitting to a single set of simulation parameters. We also utilize random disturbance forces. To further reduce the sim-to-real gap, we add actuator models [Grandia et al., 2024] to the simulator.

[1]The policy here is the same as presented in Chapter 4, relying on a latent representation of the reference motion. The motion-to-latent mapping is part of the policy $\pi(a_t|s_t, m_t)$ itself.

Critic Training. After training, the parameters of the actor are frozen and the same environment is used to learn a function that predicts the performance of the actor given a motion reference. Concretely, we aim to estimate the expected discounted cumulative reward given the current motion reference,

$$v(m) = \mathbb{E}_{\substack{s_{0:\infty} \\ a_{0:\infty} \\ m_{1:\infty}}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| m_0 = m, \pi \right], \tag{5.1}$$

where the expectation is evaluated over state-action trajectories and future motion references, and $\gamma \in [0, 1]$ is the discount factor. Variable r_t is the reward at time t, for which we choose the same reward as during RL. In principle, the reward function could be altered at this stage. The estimate (5.1) is closely related to the value function used during RL,

$$v^{\mathrm{RL}}(s, m) = \mathbb{E}_{\substack{s_{1:\infty} \\ a_{0:\infty} \\ m_{1:\infty}}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, m_0 = m, \pi \right]. \tag{5.2}$$

However, the RL value function has access to the current state of the character while $v(m)$ does not. Our reward surrogate can, therefore, be understood as the *averaged* value function over the distribution of states, thus establishing a differentiable link between kinematic motion and expected reward. Due to this similarity, we refer to the reward surrogate as *critic*.

Observing that the proposed critic is an RL critic with partial observations, we apply standard value function estimation algorithms to train a network, $v^\theta(m)$, that directly approximates Eq. (5.1). We use the approach from PPO [Schulman et al., 2017], which estimates a value function target using truncated Generalized Advantage Estimation (GAE, [Schulman et al., 2016]), corresponding to a truncated TD(λ) estimate [Sutton and Barto, 2018]. Given a finite roll-out of the current policy of length T, and a current set of parameters θ, an updated value function estimate, $\hat{v}_t$, is computed as

$$\hat{v}_t = v_t^\theta + \sum_{t'=t}^{T-1} (\gamma\lambda)^{(t'-t)} \delta_{t'}, \tag{5.3}$$

where $\delta_{t'}$ is the TD error at time t', given by

$$\delta_t = r_t + \gamma v_{t+1}^\theta - v_t^\theta. \tag{5.4}$$

With the collected batch, the critic's parameters are updated according to a square loss function

$$\min_\theta \sum \| \hat{v}_t - v_t^\theta \|_2^2. \tag{5.5}$$

The training process is outlined in Alg. 1.

Algorithm 1: Critic Training.

Input: π: control policy; $\mathcal{D}$: set of target motions m

$v^\theta \leftarrow$ init MLP with θ parameters
$\mathcal{B} \leftarrow \emptyset$ init replay buffer
/* collecting trajectories */
while $\mathcal{B}$ *not full* **do**
  $m \leftarrow$ sample motion window from $\mathcal{D}$
  $s_0 \leftarrow$ set character to random pose
  $\tau \leftarrow \emptyset$ init empty trajectory
  for $t = 0, \ldots, T$ **do**
    simulate one step using $a_t \sim \pi(a_t|s_t, m_t)$
    $r_t \leftarrow$ compute reward
    record (m_t, r_t) in τ
  end
  store τ in $\mathcal{B}$
end
/* critic updates */
for *each* τ *in* $\mathcal{B}$ **do** ▷ batch processing
  for *each* $(m_t, r_t)_{t=0}^{T}$ *in* τ **do**
    $\hat{v}_t \leftarrow$ compute value function estimate ▷ (5.3), (5.4)
  end
  $\theta \leftarrow \theta - \eta_c \nabla_\theta \left(\sum_{t=0}^{T-1} (\hat{v}_t - v_t^\theta(m_t))^2 \right)$ ▷ (5.5)
end

5.2.2 Physics-Aligned Generative Model

Training the generative model consists of a kinematic pre-training step, followed by fine-tuning. Before discussing our fine-tuning, we briefly recap the training of the generative model, which is a text-conditioned diffusion model in our case.

Denoising Diffusion Probabilistic Model. The diffusion process begins with a clean motion sequence, denoted as M_0, and progressively adds noise, resulting in a noisy sequence M_d at each step d. This process can be mathematically expressed as $q(M_d|M_0) = \mathcal{N}(M_d; \sqrt{\alpha_d} M_0, (1 - \alpha_d) I)$, with α_d representing a noise schedule that determines the intensity of the added noise [Ho et al., 2020]. Essentially, the diffusion process creates a process from clean motion to increasingly distorted motion. The objective of the

motion diffusion model is to learn the reverse process: how to denoise a sequence and gradually reconstruct the clean motion from noisy states. This is done by training the model to predict the clean motion M_0 using a parameterized function $p^\phi(M_d, d, c)$,

$$\mathcal{L}_{MDM} = \|M_0 - p^\phi(M_d, d, c)\|_2^2, \tag{5.6}$$

where c represents additional conditions like text prompts or other contextual information. By providing these conditions, the model can generate specific types of motions. This loss is minimized on randomly sampled motion-context pairs and random diffusion steps d. During inference, a random noise motion is sampled from a standard Gaussian distribution $M_D \sim \mathcal{N}(0, I)$, and D diffusion steps are applied to generate a clean motion M_0. Note that this process is not aware of physical properties that would be needed for true-to-life motion simulation.

RobotMDM. Given a pre-trained motion diffusion model and a critic, we propose to use the critic as an additional loss to fine-tune the diffusion model. We therefore generate a motion $M = p^\phi(M_d, d, c)$ and use the critic, with frozen parameters, to evaluate the expected performance for that motion. We maintain the standard MDM loss functions to ensure the model generates motions according to the data distribution and textual conditioning. However, we now also use the negative sum of critic values to indicate feasibility

$$\mathcal{L}_{RobotMDM} = \mathcal{L}_{MDM} - \beta \sum_{t=0}^{|M|} v^\theta(m_t), \tag{5.7}$$

where we sum over all the motion windows m_t contained in the generated motion. With this loss function, the MDM is trained to shape motions into more realistic examples without losing contextual accuracy, achieving higher critic values, which indicate that the policy can track the motion more accurately.

5.3 Evaluation and Results

Character. We evaluate our method on a bipedal robot with 20 degrees of freedom. The robot stands 0.84 m tall and has a mass of 16.2 kg. In simulation, we operate on a torque-controlled system [Grandia et al., 2024]; the policy outputs actuator positions that serve as inputs for the proportional-derivative (PD) controllers at each joint. We built a physical replication of

the robot where the two legs, each with 5 DoFs, are equipped with Unitree A1 actuators, while its neck and arms use Dynamixel XH540-V150-R actuators. We estimate the robot's state by using input from an onboard IMU and a motion capture setup.

Data. In this chapter, we use the textually-annotated AMASS subset [Mahmood et al., 2019] of the HumanML3D dataset [Guo et al., 2022]. This dataset is a collection of human mocap data. To retarget the motions to our robotic character, we use the inverse kinematic formulation by Schumacher *et al*. [2021]. After removing motions shorter than two seconds and mirroring them, we end up with 27112 motions annotated with 70958 textual descriptions and a total length of $\sim$55 h. We use the same train-test split as HumanML3D. Note that after retargeting, the dataset necessarily introduces artifacts due to the mismatch in topology and degrees of freedom between the SMPL body model [Loper et al., 2015] and our character.

Training Details. Tab. 5.1 provides a summary of the most important training parameters used for critic training and our physical alignment. For the pre-trained actor, we use parameters reported in Chapter 4, Tab. 4.1. During critic training, we use a fixed learning rate η_c and reduce γ to 0.9 to focus on short- and medium-term rewards. The critic is a small MLP with 3 hidden layers of size 256. As the generative backbone, we train MDM [Tevet et al., 2023] on the retargeted dataset for 3 million steps. To stabilize the training, we apply Exponential Moving Averaging (EMA, [Kingma and Ba, 2014]) over the model weights, following the implementation of Nichol *et al*. [2021]. The original MDM used 1000 diffusion steps (referred to as MDM-1K). With EMA, comparable results can be achieved in just 50 diffusion steps (referred to as MDM) [Karunratanakul et al., 2024]. We use a single EMA rate of 0.9999. The model parameters and loss function remain as reported in [Tevet et al., 2023]. RobotMDM is fine-tuned for an additional 400k steps, equivalent to 12 hours of training, using objective (5.7) and learning rate η_f.

5.3.1 Kinematic Motion Generation

We first evaluate the motion generation capability of the aligned RobotMDM, compared to the baseline MDM and PhysDiff. The results are summarized in Tab. 5.2, where the first row evaluates the performance metrics on the dataset itself. For PhysDiff, we perform a single projection step. We note that in the original PhysDiff method, the controller used during projection applies external forces to the root of the character. Given the goal of

Table 5.1: *Training Parameters.*

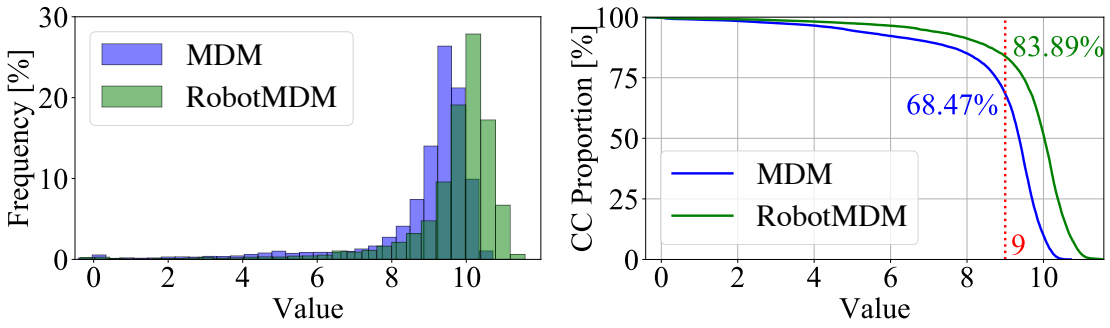
Critic Training		Physical Alignment	
Parameter	Value	Parameter	Value
Batch size	8192×32	Batch size	256
Layers	3×256	EMA rate	0.9999
η_c	$3 \cdot 10^{-4}$	η_f	0.003
γ	0.9	Fine-tuning steps	400k
λ	0.95	β	0.001
		D	50

Table 5.2: ***Kinematic Motion Generation.*** *Comparative evaluation of various kinematic motion generation methods across multiple metrics for quality, diversity, and feasibility.* ***Best*** *and second best (excluding the dataset itself).* $\pm$ *indicates the 95% confidence interval.*

Method	R-Prec. ↑	FID ↓	MM. Dist ↓	Div. →	MM. ↑	Realism ↑
Dataset	$0.696^{\pm.003}$	$0.002^{\pm.000}$	$3.799^{\pm.014}$	$8.958^{\pm.102}$	-	$6.774^{\pm.002}$
MDM-1K	$0.675^{\pm.013}$	$0.688^{\pm.090}$	$3.840^{\pm.039}$	$\mathbf{8.952}^{\pm\mathbf{.060}}$	$\mathbf{2.355}^{\pm\mathbf{.148}}$	$8.392^{\pm.036}$
MDM	$0.680^{\pm.008}$	$\mathbf{0.415}^{\pm\mathbf{.045}}$	$\mathbf{3.831}^{\pm\mathbf{.028}}$	$9.074^{\pm.135}$	$2.068^{\pm.067}$	$8.730^{\pm.018}$
PhysDiff (1-step)	$0.482^{\pm.007}$	$10.401^{\pm.089}$	$5.500^{\pm.025}$	$6.546^{\pm.037}$	$1.890^{\pm.113}$	$8.951^{\pm.034}$
RobotMDM (ours)	$\mathbf{0.684}^{\pm\mathbf{.007}}$	$0.472^{\pm.023}$	$3.835^{\pm.020}$	$9.170^{\pm.064}$	$2.087^{\pm.101}$	$\mathbf{9.562}^{\pm\mathbf{.017}}$

↑: higher is better; ↓: lower is better; →: closer to dataset is better.

deploying the motion on a real robot, we use a controller without external forces in our evaluation of PhysDiff. To evaluate the motion quality and diversity, different metrics were proposed in previous work [Guo et al., 2022; Guo et al., 2020]: the *Frèchet Inception Distance* (FID [Heusel et al., 2017]) measures the disparity between the feature distribution of the dataset and generated motions by utilizing an inception network. *R-Precision (R-Prec.)* compares the ground truth text description and 31 random text descriptions by measuring the Euclidean distance between the text embedding and the generated motion embedding. Top-3 accuracy is reported. *MultiModal dist. (MM dist.)* evaluates mode coverage by measuring the Euclidean distance between motion features and text features. To evaluate *Diversity (Div.)*, we compare the variance between the generated motions and the original motions and rank the methods based on their closeness to the dataset variance. The *MultiModality (MM.)* measures the diversity (variance) of generated motions, conditioned on the same text prompt. Finally, we propose the *Realism*



(a) *Realism Distribution.* **(b)** *CCDF.*

Figure 5.3: ***Motion Realism Comparison.*** *Comparison of MDM and RobotMDM methods for 10000 randomly-generated motions.* ***(a)*** *Distribution of Realism scores. RobotMDM shows a shift towards higher values, indicating improvements in feasibility.* ***(b)*** *Complementary Cumulative Distribution Function (CCDF) of the motion Realism values shown in (a). RobotMDM motions demonstrate significantly higher values. Anecdotally, values above 9.0 correspond to well-tracked motions.*

score, which reports the accumulated value estimated by the critic network, $\sum_{t=0}^{|M|} v^{\theta}(m_t)$, which can be taken to indicate the feasibility of the motion.

RobotMDM significantly improves the Realism score while preserving similar levels of performance across other metrics compared to MDM. Note that the dataset itself has a much lower Realism score, primarily due to noise and retargeting artifacts arising from the discrepancy between our character and the human skeleton. Compared to the dataset, motions generated by MDM are smoother and have fewer artifacts, which consequently leads to higher Realism scores. Although PhysDiff enhances the Realism of generated motions, the use of a projection using the simulated control policy results in a decreased performance in other metrics. While PhysDiff effectively eliminates ground penetrations, it heavily depends on the controller's tracking accuracy. We hypothesize that the absence of external helper forces in our control policy necessitates large projection steps and reduces the effectiveness of this projection strategy, ultimately leading to less versatility. Rather than projecting the motions, RobotMDM learns a strategy to circumvent infeasible motions, thereby sustaining quality and diversity. Additionally, in contrast to PhysDiff, RobotMDM does not add any extra computational overhead to MDM during motion generation.

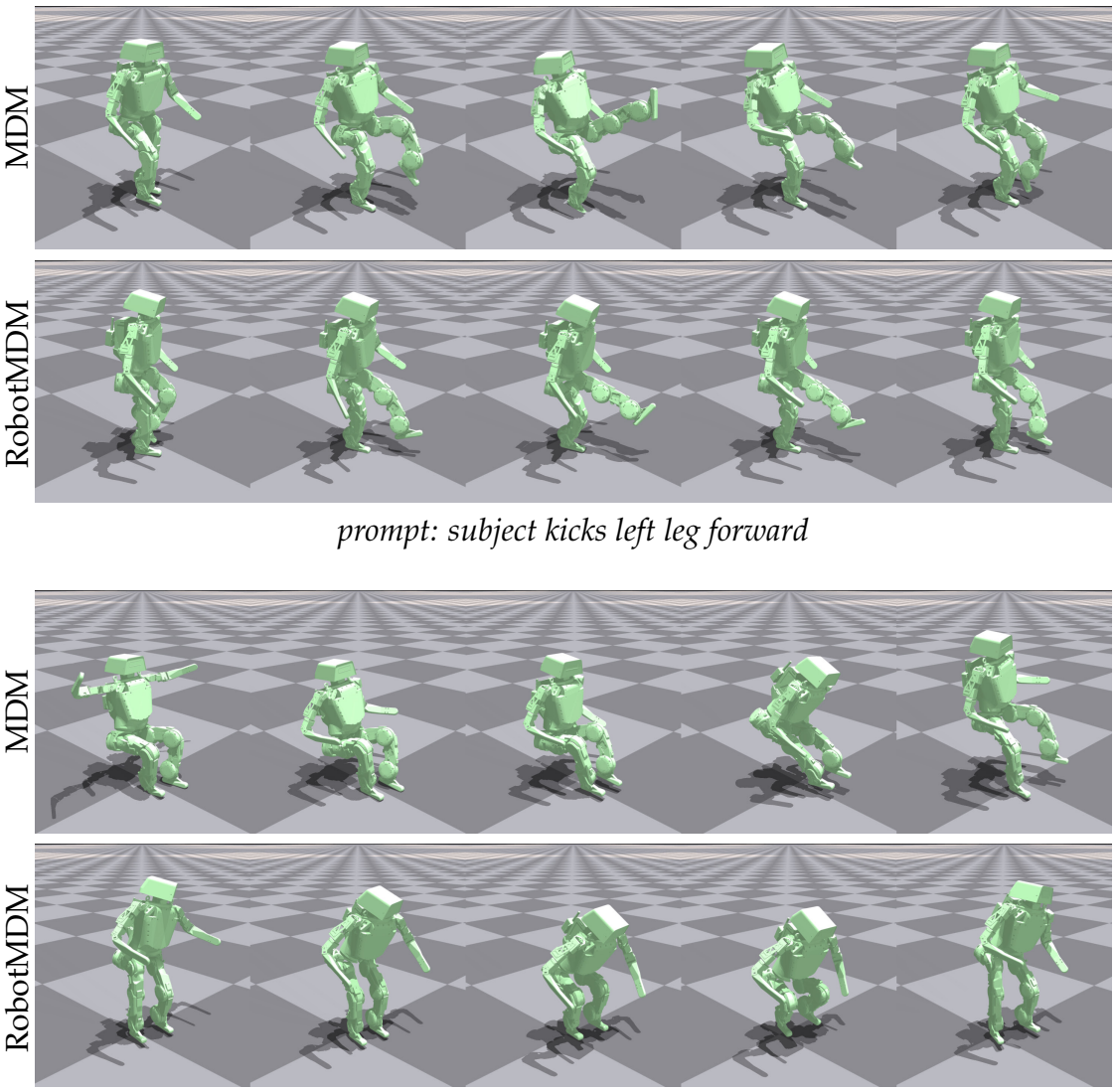


Figure 5.4: ***Realistic Motion Generation.*** *Aligning the motion diffusion model with physical knowledge results in more realistic motions within the character's limits while preserving the context. This results in a less extreme kick where the character also remains more balanced, or a sitting motion that is feasible in the absence of a chair.*

5.3.2 Physical Alignment

In the following, we compare generated motions, with a focus on feasibility. The dataset includes motions involving object interactions, such as sitting on a chair, which are generally not feasible without the presence of the object. Additionally, the dataset was collected from human subjects, and kinematically retargeting these motions to our character does not guarantee that the character can perform them. Therefore, simply learning the motion distribution from the dataset is insufficient for achieving physical realism.

Fig. 5.4 shows two examples where RobotMDM refines the motions to enhance realism. The first example depicts a leg kick where the MDM-generated motion is imbalanced and features an excessively strong high kick. Given the character's inability to capture such dynamic motions, our method moderates the strength and stabilizes the movement, making it more realistic. The second example in Fig. 5.4 demonstrates a motion involving sitting. In MDM, the character appears to sit in mid-air, as similar motions are present in the dataset. Our method recognizes the infeasibility of this action and adjusts accordingly, resulting in a version where the character squats down instead of leaning back.

Another issue, resulting from the retargeting process, is that the dataset contains motions with self-collisions. MDM, therefore, produces motions where body parts intersect. Such intersections lead to a lower reward, as the policy will not be able to track them accurately. Consequently, the fine-tuned RobotMDM avoids these intersections. Fig. 5.5 visualizes the character's collision bodies during a waving motion, where bodies are colored red during collisions (*e.g.*, when the feet contact the floor). MDM results in a collision between the head and arm, whereas RobotMDM successfully avoids such issues while preserving the expressiveness of the motion.

5.3.3 Physics-Based Motion Tracking

Next, we assess the tracking performance of the control policy. A total of 10000 motions, each 10 s long, are generated for each method based on test prompts, resulting in 30 hours of motion. We evaluate the performance using 2048 simulation episodes, each lasting 30 s. During these episodes, the policy attempts to imitate the target motions, starting from a randomly selected frame. If the selected motion ends or the character is terminated, a new motion is sampled. We summarize tracking performance results in Tab. 5.3. Poses generated by RobotMDM are tracked with greater accuracy, particularly the lower body pose. Motions generated by MDM are more

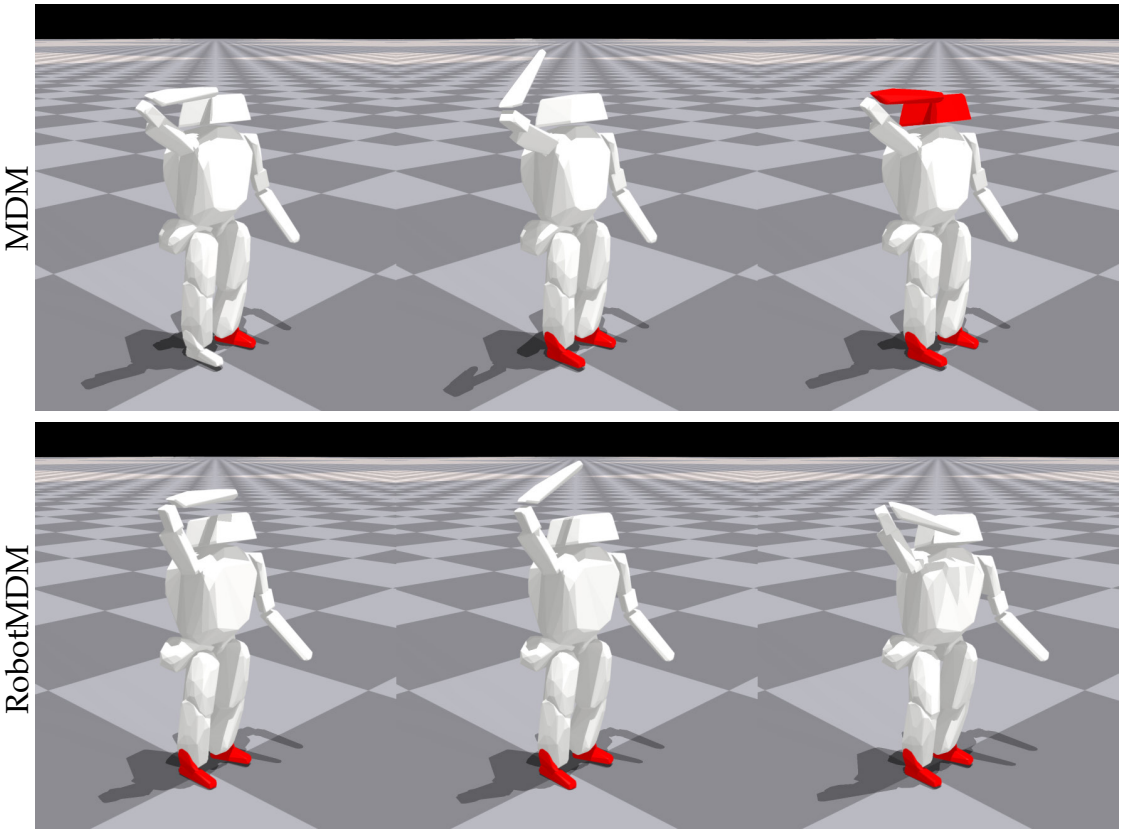


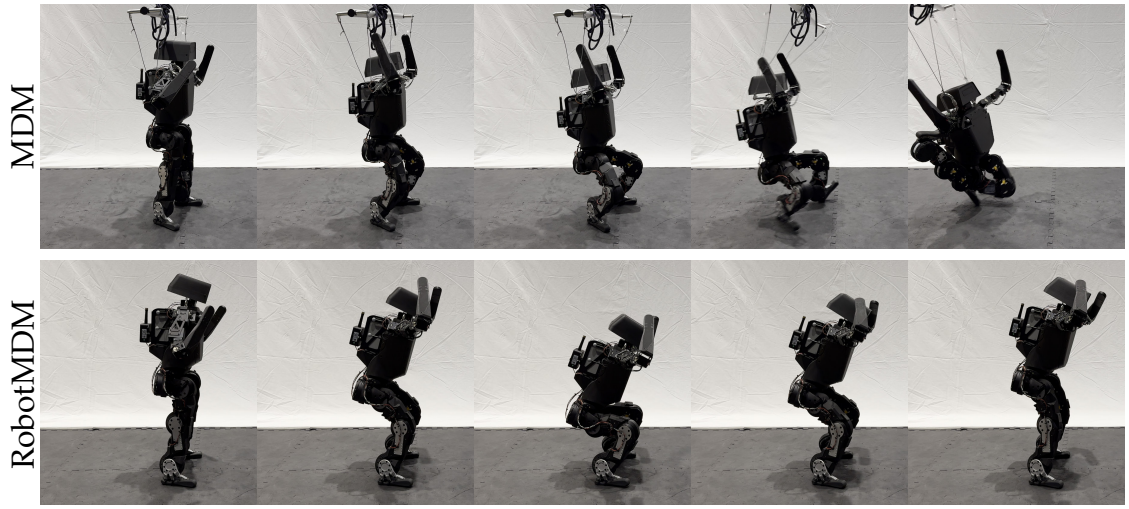
Figure 5.5: ***Collision Avoidance.*** *Collisions between bodies result in a lower reward, because they are not accurately tracked by the policy. The aligned RobotMDM naturally circumvents collisions.*

frequently infeasible, often featuring overly expressive leg movements. Additionally, root rotation errors are nearly halved as the motions created by RobotMDM are more balanced, requiring smaller corrections to the root pose. Furthermore, both the linear and angular velocities are more closely aligned with what is feasible on the robot, enhancing the overall realism and functionality of the generated motions. Figs. 5.1 and 5.7 show the tracking of expressive motions on the real robot, and additional results are provided in the supplementary video.

The ability to generate motions can also be leveraged to enhance the motion tracking policy. While the original tracking policy was trained on the retargeted dataset, we retrain the policy based on the generated motions from MDM and RobotMDM, resulting in a tailored tracking policy for each motion generator. We find that even after specializing the policy on the motions generated by MDM, the generated motions remain hard to execute stably, confirming that the motions are indeed infeasible. Fig. 5.6 presents an

Table 5.3: ***Generated Motion Tracking.*** *Evaluation of tracking performance across linear and angular root velocity, root rotation, and upper and lower body Degrees of Freedom (DoFs) tracking, measured over 2048 simulations of 30-second references from motions generated by MDM and RobotMDM.*

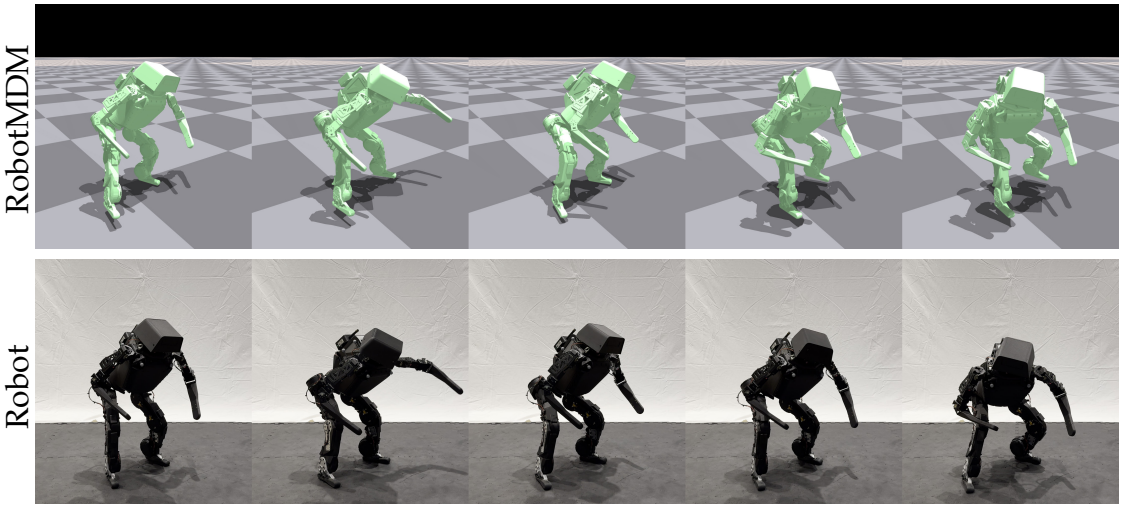
	Tracking Error				
Input	lin. vel. [m/s]	ang. vel. [rad/s]	root rot. [°]	upper DoFs [°]	lower DoFs [°]
MDM	4.90	0.29	4.13	10.88	16.11
RobotMDM	3.43	0.23	2.34	9.36	11.44



prompt: the person lifts a dumbbell over his head.

Figure 5.6: ***Robot Control.*** *MDM motions are difficult to track on a real robot system—the lack of balance and non-physicality lead to poor target matching. The same prompt for RobotMDM yields better robot motions.*

example of the post-training performance: The character performs a lifting motion. RobotMDM-generated motions for this prompt are feasible, while those from MDM cause the robot to lose balance. See the supplementary video for similar results. The clear differences between the two types of generated motions underscore the potential benefits of iteratively improving both the control policy and the motion generator. By enhancing these components in tandem, we can further boost the precision and reliability of the robot's performance.



prompt: a person sneakily crouches while moving laterally.

Figure 5.7: ***Robot Precision.*** *The generated motions can be accurately tracked on a real-world robot.*

5.4 Concluding Remarks

Contributions. In this chapter, we combine kinematic motion generation with physics-based character control. Our method aligns a motion diffusion model with the physical constraints of the character without compromising versatility and diversity, and can be deployed on real-world robots.

Limitations. While the generated motions align more closely with the desired objectives, there is no hard constraint on motion feasibility. Prompts that significantly deviate from what the character can realistically do might require substantial adjustment of the motion to become feasible, leading to a conflict with the original text prompt. As shown in the video, when prompted to make the robot swim, RobotMDM struggles to resolve this conflict completely. We can, therefore, not guarantee its reliability in performance-critical environments, as there remains a risk that the model generates motions that violate constraints or are infeasible. Finally, despite significant advancements in the past year, the need for a character-specific dataset, often derived from human data, does not fully capture the expressiveness that robotic hardware could support. We believe that further bridging the gap between kinematic and physics-based approaches through large-scale synthetic datasets generated from physics-aware motion generators will ultimately lead to enhanced policies and robot capabilities.

CHAPTER 6

Conclusion

We have pushed the boundaries of dynamic motions for robotic characters, enhancing their capabilities of performing motions in the real world that look and feel lifelike. The effect such motions have on how humans perceive the robot is remarkable. The difference between a static idle pose and one with subtle, human-like breathing motions makes a character feel significantly more believable and alive.

In our work, we have leveraged the strengths of different fields to develop new solutions that advance the state-of-the-art. We combined simulators with neural networks, used the insights from animation pipelines to add splines into transformers, combined pretrained latent spaces obtained through self-supervised learning with control policies using reinforcement learning, and connected kinematic motion generation with physics-based character control.

In the following, we summarize the key contributions of our work and discuss its limitations. We conclude the thesis with an outlook on future directions, outlining immediate research opportunities and presenting speculative long-term visions for the next generation of research.

6.1 Summary and Limitations

Chapter 2 presents a neural approach to augment robot simulation states. In contrast to previous methods, we propose to learn separate neural models for different robot building blocks. To achieve this modular augmentation, we decouple the building block by considering the constraint forces

and torques that act between the components. The neural modules then learn from real-world data, significantly reducing the sim-to-real gap and thereby increasing simulation accuracy. The modular structure of the formulation allows for zero-shot transfer to new robots built from similar components, without having to recollect data or retrain new models. However, the generalization capabilities have limitations; while we observe good generalization across robots for actuators, generalization across rigid bodies is restricted. Robust rigid body generalization can only be achieved for variations in physical parameters, such as the mass, but does not extend across unseen rigid bodies. To achieve a better generalization, we hypothesize that a larger dataset with diverse rigid bodies has to be collected. Although we later observe that even inaccurate simulation of robots in combination with domain randomization, disturbances, and noisy states can successfully lead to controllers that transfer to robots, we suspect that neural-based simulations with higher simulation accuracy could improve fine motor skills, such as grasping, facial actuation or maintaining unstable states over and extended period, *e.g.*, balancing on one leg.

Chapter 3 is driven by the observation that motions are usually represented as spline curves in computer animation, and so should their latent representation be. We designed a new model called the Spline-based Transformer. This architecture encodes sequential data as a trajectory in latent space, eliminating the need for any explicit positional encoding. In Chapter 3, we show that Spline-based Transformers outperform their traditional counterparts across a range of data modalities and achieve higher reconstruction quality. Furthermore, the spline-based latent space offers a new way to control latent spaces showing interesting, non-linear behavior. While promising, the model is still in its early stages and has yet to demonstrate its effectiveness in more practical applications.

Chapter 4 introduces a two-stage approach that combines a latent space with a controller to accurately track motions not only on simulated characters but also on robots. In this chapter, we demonstrate that a policy trained using reinforcement learning can benefit from a pretrained encoder and achieves higher tracking accuracy, generalizes better, and converges more efficiently. We hypothesize that the latent codes enable the policy to identify similarities between motion segments, resulting in more general learning and better transfer to unseen motions. Improving the quality of the latent space alone leads to higher tracking accuracy, further supporting this hypothesis. While this approach demonstrates a wide range of motions in both simulation and the real world, it still faces limitations. We observe that, especially for motions that require longer horizon planning, the accuracy decreases. Furthermore, since we directly condition on reference motions and apply explicit

reward terms, the pipeline is highly dependent on the quality of the motion dataset. While our method achieves high-quality results in the real world, it currently relies on a motion capture system to improve the state estimation of the robot. Deploying similar dynamic motions in non-studio settings will require more accurate state estimation methods or higher robustness of the control policy.

Chapter 5 presents RobotMDM, a motion generator that produces motions that better align with the limitations of the character and the capabilities of the controller. We observed that current motion generators produce motions that visually look appealing but can not reliably be tracked on the robot. To improve this gen-to-real gap, we propose to use the *critic* obtained during reinforcement learning as an additional loss term to finetune the motion generator. The critic acts as a surrogate for physics and estimates the tracking accuracy of the policy when conditioned on a kinematic reference motion. While effective, the method provides no guarantees and can still lead to generated motions that are not feasible.

6.2 Future Research

Robotic characters are at a pivotal moment, receiving increasing attention and a growing interest, and offer many promising directions for future research. Despite the current progress, the field still faces numerous open challenges and demands substantial work in many different areas. At the center of many problems are the motion capabilities of robots. The motions robots perform do not always have to be lifelike; more often, such as in factories, we only require a functional behavior. However, once robots operate in environments with direct or indirect interactions with humans, lifelike motions become highly relevant. With more believable characters, humans are not only more entertained but also feel safer, as such motions are better to predict and understand. This is essential in a future where we collaborate on tasks or have robots in our homes.

While hardware still remains a significant factor that explicitly limits what a robot can do, we believe there is still a lot of room for improvement by developing more effective methods and strategies.

We will first discuss future work that can have a direct impact on further improving the work presented in this thesis, and will increasingly open up the scope towards more speculative and broader research directions.

Retargeting. Since we can not put our robots into motion capture suits and let them perform the motion we want to learn—chicken-and-egg problem—we have to perform retargeting of human motions to our robotic characters. This step is important and has an impact on the entire work of this thesis. The retargeting problem is ill-posed, and there is not always a one-to-one mapping between a human and a robotic character. Even if the robot and human share the same morphology, differences in size and ratios of limbs can make this problem challenging. While there is active work in this direction for virtual characters [Lee et al., 2023], the focus is often on achieving visually appealing results. However, the retargeting problem has large dynamic challenges that remain difficult to address—how to adapt motion velocity, how to maintain dynamic balancing, or how to account for physical constraints and limitations. We hope to see more methods explicitly tailored for real-world robotics.

Implicit Rewards. Another important direction for future work is advancing reward design—how to combine multiple components and how to formulate different terms. During this work, we spent a lot of time manually tuning reward weights and adjusting their formulation. In a recent work, we investigated a multi-objective formulation where the reward weights can be changed in post-training, allowing us to even tune the reward on the real system [Alegre et al., 2025]. While promising, this only solves part of the problem and still requires a predefined formulation of the reward terms. Adversarial Motion Priors are an interesting direction [Peng et al., 2021]. Here, the reward is modeled implicitly through a discriminator model, which tries to distinguish between motions from the dataset and the motions performed by the policy. However, current challenges, including mode collapse and slower training convergence, have limited their adaptation. And the latest large-scale controllers still rely on explicit formulations of the reward [Tessler et al., 2024]. We hope to see advances in implicit rewards for large-scale imitation tasks.

Generative Control. RobotMDM, presented in Chapter 5, combines motion diffusion models with motion trackers. While this work finetunes the generator, more recent work tunes the controller instead and provides a closed-loop integration between generator and tracker [Tevet et al., 2025]. We believe that an even tighter connection between the two models, established during training, can lead to more powerful methods for creating stable controllers with strong generative capabilities.

Lifelike Behavior. To make a robot lifelike, the context in which a motion is performed also has to match our expectations. Ultimately, we aim to develop robots that perceive the world, traverse it, and react autonomously within it. This goal is multi-disciplinary and requires expertise in vision, text, and control. Robotic systems are making great progress in this direction, especially in autonomously traversing the world [Miki et al., 2022]. To achieve autonomous interaction with humans, we introduced a system that learns to control a robot by imitating an expert operator while performing a human-robot interaction [Christen et al., 2025]. Again, this method makes use of multiple components: a diffusion model learns to imitate the operator, while the commands are streamed to a pretrained policy that executes the motion on the robot. Incorporating a broader understanding of the interaction with humans and the environment is an exciting future avenue. Recently introduced Vision-Language-Action models are a promising effort in this direction of achieving general autonomous robot capabilities [Bjorck et al., 2025; Team et al., 2025].

Control Complexity. As freely walking robots become more complex and their actuation space increases, the control problem becomes increasingly more challenging. The human body has over 200 degrees of freedom (DOF) and more than 600 skeletal muscles; the hands alone have 54 DOF, and their control is a whole research field on its own [Christen et al., 2022]. Yet, despite this complexity, humans perform coordinated motions with remarkable fluidity and efficiency, and mostly without conscious effort. Low-level motor control systems take care of most of the motion we perform by leveraging learned motor patterns and reflexes. Designing similar systems to replicate this level of coordination in robotics remains a fundamental challenge that requires further advances in control architectures and learning algorithms.

Unconstrained Learning. Data is the strictest constraint of the current learning systems. There is only so much information we can extract from it, and to learn more and advance the state, we have to be able to reason, to order thoughts, and to *experience*, which is why reinforcement learning is such a fascinating field. Here, intelligence evolves by trying out, *explore*, and diving in, *exploit*, the learnings. For robots, the simulated worlds are currently a limit; the experience a robot can acquire in those is predefined, and it sets a hard boundary on learning. As robots continue to advance in their ability to navigate our world, we must explore new learning methods suited to open environments—and ensure that their behavior remains aligned with ours.

APPENDIX A

Additional Details: Spline-based Transformers

A.1 More About Control Points

Role In Downstream Applications. We evaluate Spline-based Transformers mainly from the point of view of reconstruction tasks as they are a common proxy task for pre-training feature backbones for various downstream applications (such as classification, segmentation, etc). Even when trained to reconstruct input sequences, Spline-based Transformers already seem to capture the similarities and differences between the input sequences in their latent control points. In Fig. A.1, we show the predicted control points of two similar-looking curves (Cols 1, 2) alongside a different curve (Col 3) with their corresponding latent trajectories in the second row. Complex curves appear to have more nonlinear latent trajectories (Cols 1, 2), while the smoother curve (Col 3) has a more linear latent trajectory. While further experimentation is certainly required, our early results indicate that Spline-based Transformers will find use in various other applications, including representation learning, classification *etc*.

Latent Control Manipulation. The number of control tokens/points depends on the type of B-Spline (cardinal B-Splines, cubic Bézier, etc.) used. So far we have used cubic Bézier splines, which are defined by four control tokens across all data modalities. These control points determine the trajectory of the input sequence in the latent space of a Spline-based Transformer. For cubic Bézier splines, the first and the last latent control points determine

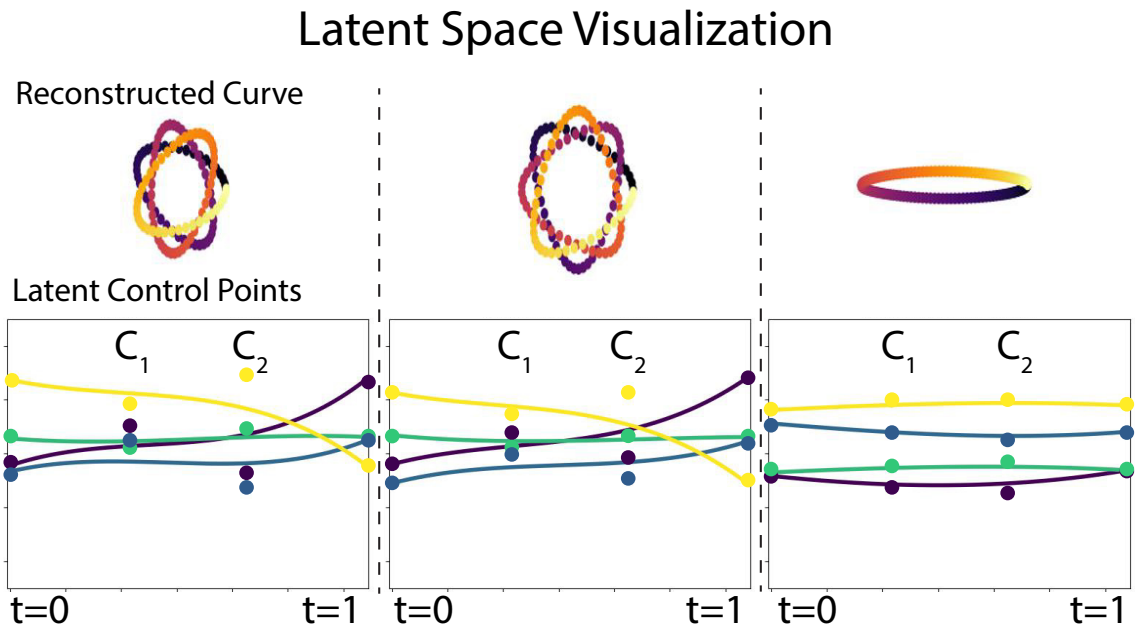


Figure A.1: ***Latent Space Similarities.*** *We show Hypotrochoid curves generated by our method along with their corresponding latent control points (shown as dots) and the latent trajectories obtained using a cubic Bézier interpolator. Curves that look similar in Cartesian space (Cols 1, 2) seem to have similar latent controls and trajectories, while smoother curves (Col 3) seem to have smoother latent trajectories.*

the start and end of the latent trajectory, while the second C_1 and third C_2 control points define the shape of the latent spline. Fig. A.2 demonstrates the effect of modifying the control point C_2 on 2D Hypotrochoids and the effect this has on reconstruction.

A.2 Implementation Details

In the following, we present the implementation details for our Spline-based Transformer to help with reproducibility. In Tab. A.1, we enumerate the various hyperparameters used in building the Spline-based Transformer for each of our experiments.

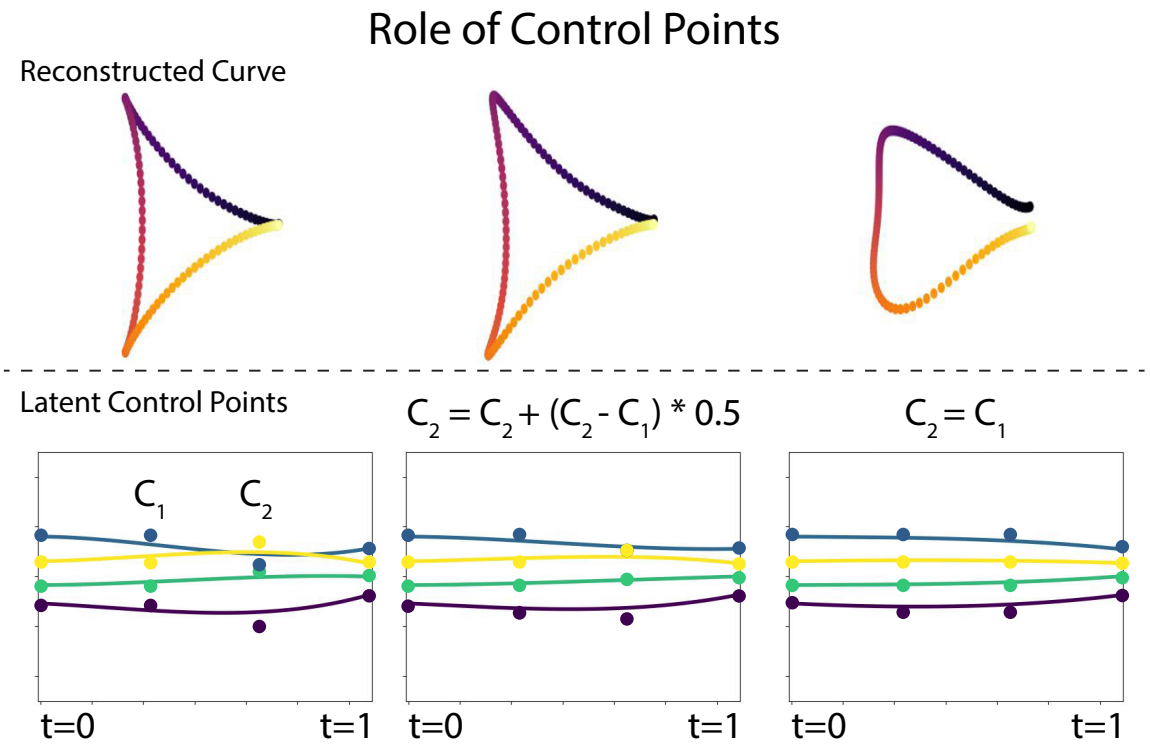


Figure A.2: ***Modifying Control Points.*** *Modifying the control points alters the latent trajectories according to the chosen B-Spline (cubic Bézier in our case). Here we incrementally modify control point C_2 to be closer to C_1 and visualize the change in the latent space. The corresponding reconstructed curve is shown in the top row.*

Algorithm 2: Spline-based Transformer.

Input: Sequence x of dimensions $(seq\_len \times data\_dim)$.
Param: Number of control points *controls*. Dimension of spline d.
Network layers {encoder,decoder,T5}_layers.
Output: Predicted sequence $\tilde{x}$.

```
/* components                                                    */
MLPEnc = Sequential(encoder_layers)
MLPDec = Sequential(decoder_layers)
TEnc = TransformerEncoder(T5_layers)
TDec = TransformerDecoder(T5_layers)
p = ParameterList([Parameter(d) for _ in range(controls)])
/* forward pass                                                  */
feat_in = MLPEnc(x)
enc_inp = cat(*p, feat_in)                          ▷ concat control points
enc_out = TEnc(enc_inp)

c_points = enc_out[:controls]
latents = EVALUATE(c_points, seq_len)                           ▷ Eq. 3.1

feat_out = TDec(latents)
x̃ = MLPDec(feat_out)
```

Algorithm 3: EVALUATE CubicBézier

Input: Four control points *c_points* and sequence length *seq_len*.
Output: Uniformly evaluated latent spline **s**.

```
1 t = linspace(0, 1, seq_len)
2 s = ((1 − t)^3 · c_points[0]
3       + 3.0 · (1 − t)^2 · t · c_points[1]
4       + 3.0 · (1 − t) · t^2 · c_points[2]
5       + t^3 · c_points[3])
6 return s
```

Spline-based Transformers only require simple modifications to a traditional transformer model and easily fit into various existing setups. Alg. 2 presents a general implementation of Spline-based Transformers using a torch-like syntax. Different layers can be chosen for MLP encoder/decoder, *e.g.*, in the case of image data, 2D convolution layers can be used. Similarly, different transformer blocks can be stacked together to build our transformer encoder/decoder. Our control tokens **p** are learnable parameters (Line 6) and are initialized from a normal distribution. These learnable control tokens are concatenated to the sequence of tokens after the MLP encoder (Line 9) to result in the transformer encoder's input sequence *enc_inp*. The tokens corresponding to the control tokens are interpreted as latent control points at the output of the encoder, which are then evaluated as a spline (Line 11-12). In our experiments, we used cubic Bézier splines (with four control points), and Alg. 3 shows how we evaluate them to obtain a latent token sequence for the decoder. Finally, the interpolated latents are passed through the transformer decoder and the shared MLP decoder and ultimately mapped back into their original space. An L2 loss function between the input and output sequence is used to train our system end-to-end.

Tab. A.1 summarizes the parameters and hyperparameters used to conduct the experiments presented in the paper. The latent dimension d, number of stacked transformer layers n, number of transformer heads per layer h, and the feature size of each layer c. Internally, the transformer layers consist of Feed-Forward Networks (FFN), two fully connected layers with non-linear activation GELU. FFNs can have an inner dimension that is 1-4x larger than their outer dimension. We keep the inner structure equal and set the factor to 1x. BS is the batch size, and lr is the learning rate. For the image results, we used a patch size PS. The number of parameters corresponds to the model with the largest latent dimension.

Table A.1: *Parameters and Hyperparameters.*

Sec	Experiment	Param.	d	n	h	c	FFN	BS	lr	PS
3.3.1	Lissajous (3D)	0.20M	3	4	4	64	1	256	$1e^{-3}$	-
	Hypotrochoids (4D)	0.20M	4	4	4	64	1	256	$1e^{-3}$	-
	Bézier (2D)	0.20M	2	4	4	64	1	1024	$1e^{-3}$	-
	Bézier (64D)	0.43M	64	4	4	128	1	1024	$1e^{-3}$	-
3.3.2	CIFAR10	0.85M	$2^{5,6,7}$	4	8	128	1	512	$3e^{-4}$	4
	AFHQ	1.65M	$2^{5,6,7}$	4	8	128	1	512	$3e^{-4}$	32
	Face images	1.65M	$2^{5,6,7}$	4	8	128	1	512	$3e^{-4}$	32
3.3.3	Faces	12.3M	$2^{5,6,7,8}$	4	8	256	1	32	$5e^{-5}$	-
	Motions	0.63M	$2^{4,5,6}$	4	8	128	1	1024	$3e^{-4}$	-
3.3.4	Strands	0.21M	$2^{3,4,5}$	4	8	64	1	128	$1e^{-3}$	-

A.3 Additional Results

Additional results for reconstructing synthetic curves, facial images, and hair strands are shown in Figs. A.3, A.4, and A.5, respectively.

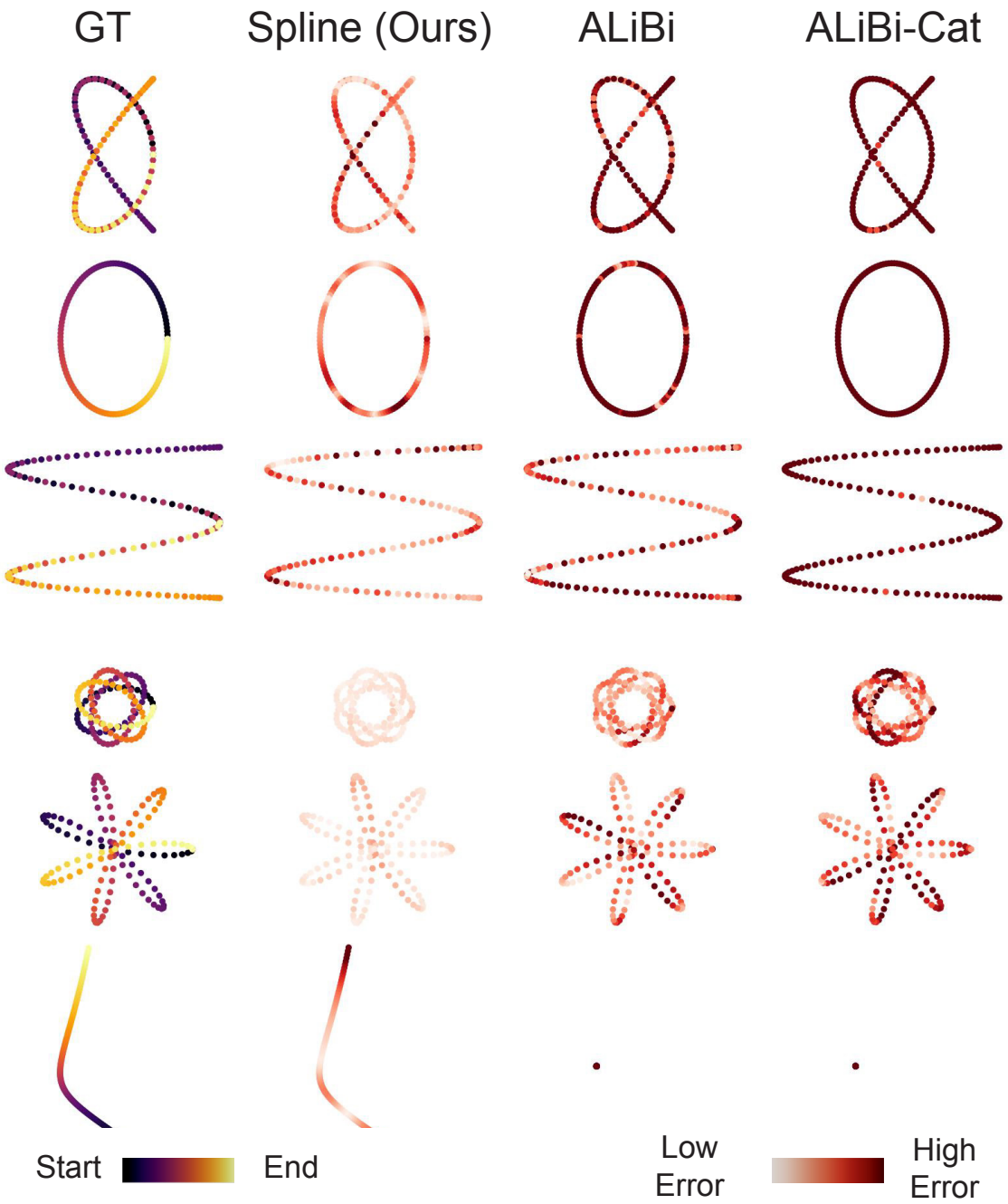


Figure A.3: ***2D Curves.*** *Additional reconstruction results for 2D curves.*

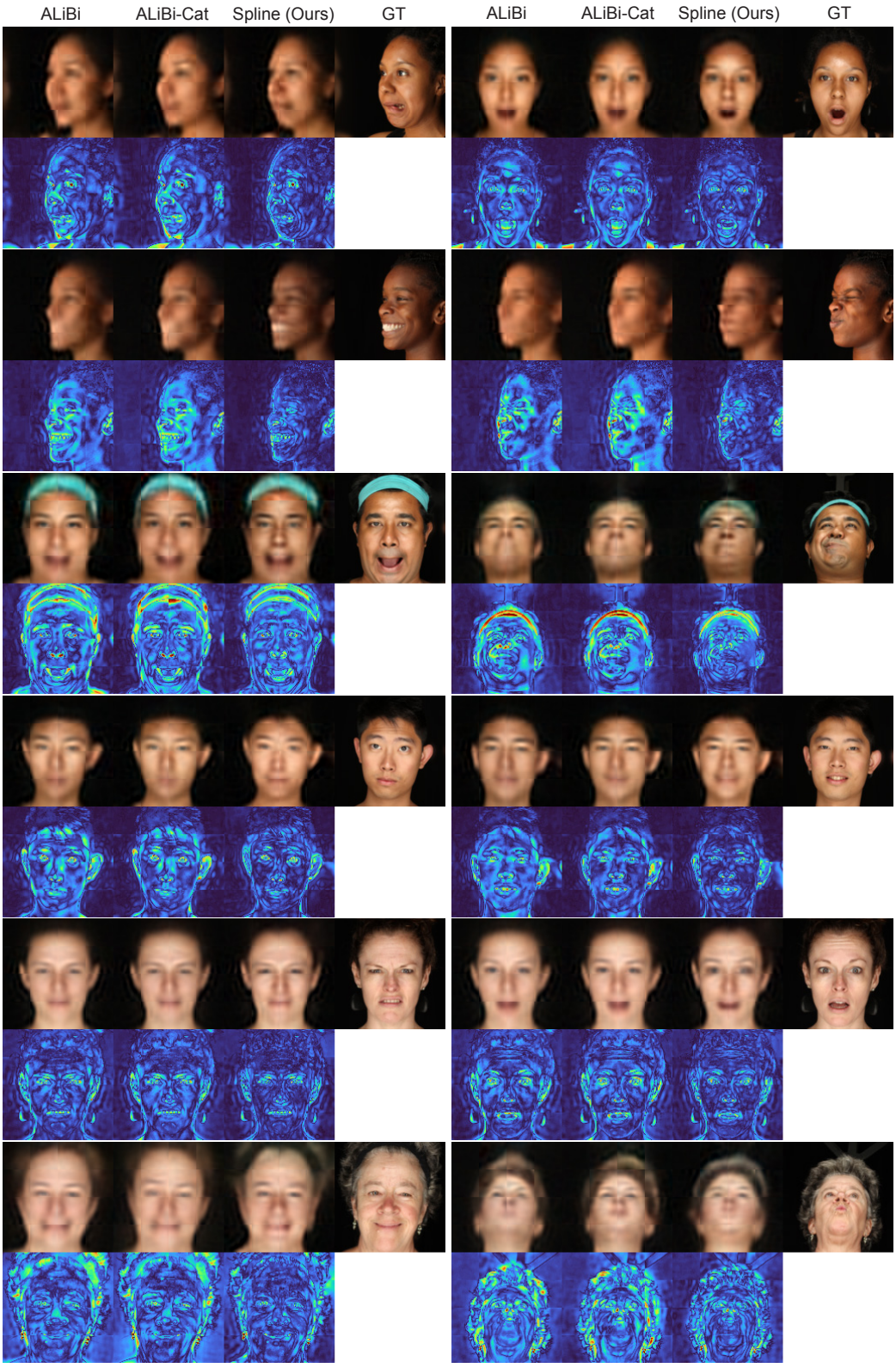


Figure A.4: *Additional reconstruction results for test images.*

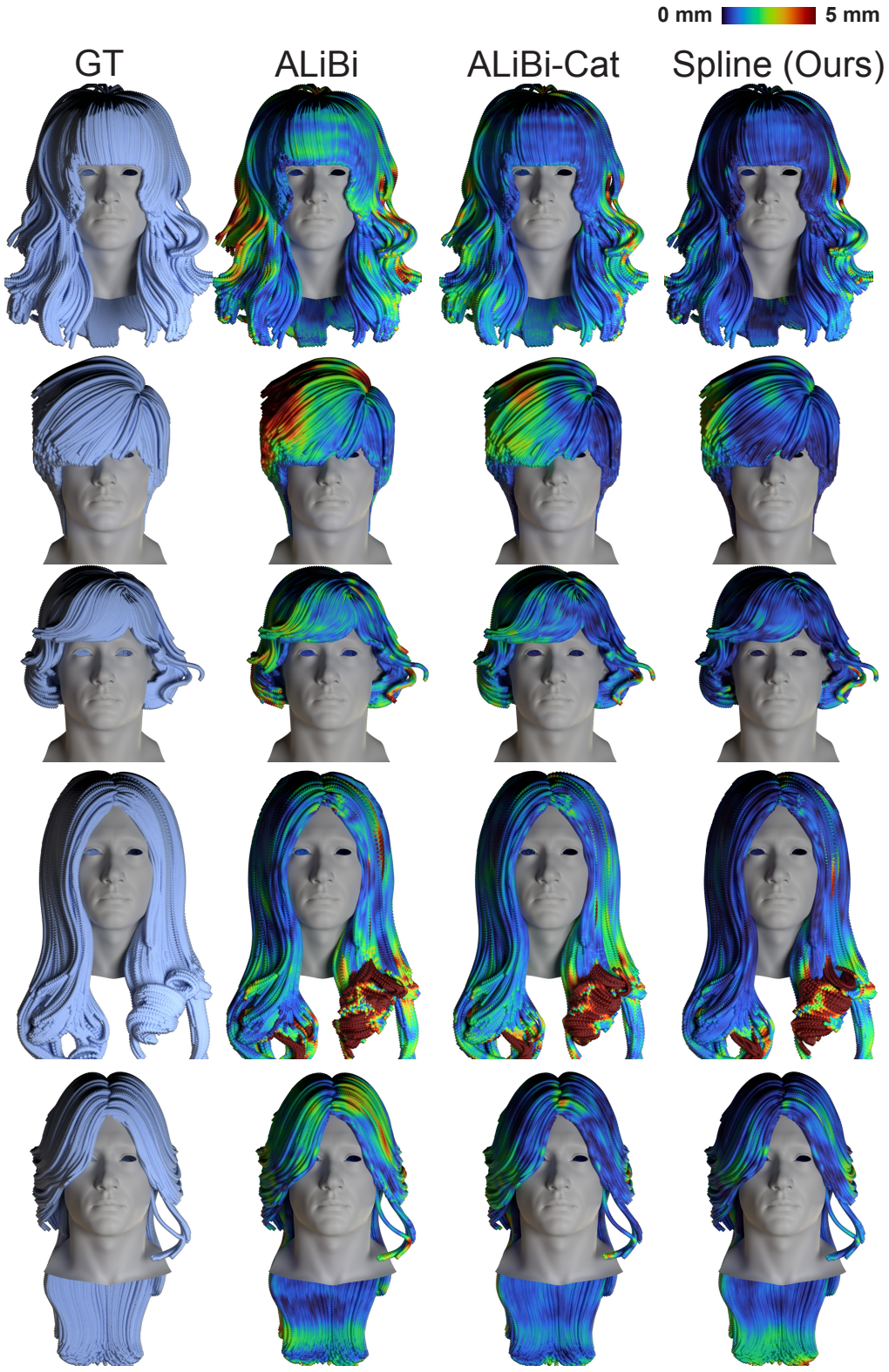


Figure A.5: ***Hairstyles.*** *Additional reconstruction results for test hairstyles.*

References

[Ajay et al., 2018] Anurag Ajay, Jiajun Wu, Nima Fazeli, Maria Bauza, Leslie P Kaelbling, Joshua B Tenenbaum, and Alberto Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *IEEE/RSJ IROS*. IEEE, 2018.

[Akiba et al., 2019] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[Al-Khateeb et al., 2023] Faisal Al-Khateeb, Nolan Dey, Daria Soboleva, and Joel Hestness. Position interpolation improves alibi extrapolation. *CoRR*, abs/2310.13017, 2023.

[Alegre et al., 2025] Lucas N. Alegre, Agon Serifi, Ruben Grandia, David Müller, Espen Knoop, and Moritz Bächer. Amor: Adaptive character control through multi-objective reinforcement learning. In *ACM SIGGRAPH 2025 Conference Proceedings*, SIGGRAPH '25, 2025.

[Aneja et al., 2023] Shivangi Aneja, Justus Thies, Angela Dai, and Matthias Nießner. Facetalk: Audio-driven motion diffusion for neural parametric head models, 2023.

[Ba et al., 2016] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[Baevski et al., 2020] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *NeurIPS*, 33, 2020.

[Beitner and colleagues, 2020] Jan Beitner and colleagues. Pytorch forecasting. `https://github.com/jdb78/pytorch-forecasting`, 2020.

[Bengio et al., 1994] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2), 1994.

[Bergamin et al., 2019] Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. Drecon: Data-driven responsive control of physics-based characters. *ACM Trans. Graph.*, 38(6), nov 2019.

[Bianchi et al., 2017] Filippo Maria Bianchi, Enrico Maiorino, Michael C Kampffmeyer, Antonello Rizzi, and Robert Jenssen. An overview and comparative analysis of recurrent neural networks for short term load forecasting. *arXiv preprint arXiv:1705.04378*, 2017.

[Bishop, 1995] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., USA, 1995.

[Bjorck et al., 2025] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[Bommasani et al., 2021] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[Brockman et al., 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[Carpentier and Mansard, 2018] Justin Carpentier and Nicolas Mansard. Analytical derivatives of rigid body dynamics algorithms. In *Robotics: Science and Systems*, 2018.

[Chandran et al., 2020] P. Chandran, D. Bradley, M. Gross, and T. Beeler. Semantic deep face models. In *2020 International Conference on 3D Vision (3DV)*, pages 345–354, Los Alamitos, CA, USA, nov 2020. IEEE Computer Society.

[Chandran et al., 2022a] Prashanth Chandran, Gaspard Zoss, Markus Gross, Paulo Gotardo, and Derek Bradley. Facial animation with disentangled identity and motion using transformers. *Computer Graphics Forum*, 41(8):267–277, 2022.

[Chandran et al., 2022b] Prashanth Chandran, Gaspard Zoss, Markus Gross, Paulo Gotardo, and Derek Bradley. Shape transformers: Topology-independent 3d shape models using transformers. In *Computer Graphics Forum*, volume 41,

pages 195–207. Wiley Online Library, 2022.

[Chandran et al., 2024] Prashanth Chandran, Agon Serifi, Markus Gross, and Moritz Bächer. Spline-based transformers. In *Computer Vision – ECCV 2024*, pages 1–17, 2024.

[Chen et al., 2023] Xin Chen, Biao Jiang, Wen Liu, Zilong Huang, Bin Fu, Tao Chen, and Gang Yu. Executing your commands via motion diffusion in latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18000–18010, 2023.

[Choi et al., 2020] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[Christen et al., 2022] Sammy Christen, Muhammed Kocabas, Emre Aksan, Jemin Hwangbo, Jie Song, and Otmar Hilliges. D-grasp: Physically plausible dynamic grasp synthesis for hand-object interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[Christen et al., 2025] Sammy Christen, David Müller, Agon Serifi, Ruben Grandia, Georg Wiedebach, Michael A. Hopkins, Espen Knoop, and Moritz Bächer. Autonomous human-robot interaction via operator imitation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025.

[Christiano et al., 2017] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[Chung et al., 2015] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *NeurIPS*, 28, 2015.

[Clevert et al., 2016] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016.

[CMU, 2001] CMU. Cmu graphics lab motion capture database, 2001.

[Daněček et al., 2023] Radek Daněček, Kiran Chhatre, Shashank Tripathi, Yandong Wen, Michael Black, and Timo Bolkart. Emotional speech-driven animation with content-emotion disentanglement. In *SIGGRAPH Asia 2023 Conference Papers*, SA '23, New York, NY, USA, 2023. Association for Computing Machinery.

[de Avila Belbute-Peres et al., 2018] Filipe de Avila Belbute-Peres, Kevin Smith,

Kelsey Allen, Josh Tenenbaum, and J Zico Kolter. End-to-end differentiable physics for learning and control. *NeurIPS*, 31, 2018.

[Devlin et al., 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, 2018.

[Dhariwal et al., 2020] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020.

[Dosovitskiy et al., 2021] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[Dou et al., 2023] Zhiyang Dou, Xuelin Chen, Qingnan Fan, Taku Komura, and Wenping Wang. C·ase: Learning conditional adversarial skill embeddings for physics-based characters. In *SIGGRAPH Asia 2023 Conference Papers*, New York, NY, USA, 2023. Association for Computing Machinery.

[Duan et al., 2021] Yinglin Duan, Tianyang Shi, Zhengxia Zou, Yenan Lin, Zhehui Qian, Bohan Zhang, and Yi Yuan. Single-shot motion completion with transformer. *arXiv preprint arXiv:2103.00776*, 2021.

[Erez et al., 2015] Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *IEEE ICRA*, 2015.

[Farin, 2001] Gerald Farin. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2001.

[Feng et al., 2023] Yusen Feng, Xiyan Xu, and Libin Liu. Musclevae: Model-based controllers of muscle-actuated characters. In *SIGGRAPH Asia 2023 Conference Papers*, SA '23, New York, NY, USA, 2023. Association for Computing Machinery.

[Fu et al., 2019] Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 240–250. Association for Computational Linguistics, June 2019.

[Fukushima, 1969] Kunihiko Fukushima. Visual feature extraction by a multilay-

ered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969.

[Fussell et al., 2021] Levi Fussell, Kevin Bergamin, and Daniel Holden. SuperTrack: motion tracking for physically simulated characters using supervised learning. *ACM Trans. Graph.*, 40(6):1–13, December 2021.

[Garcia et al., 1989] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[Gehring et al., 2023] Jonas Gehring, Deepak Gopinath, Jungdam Won, Andreas Krause, Gabriel Synnaeve, and Nicolas Usunier. Leveraging demonstrations with latent space priors. *Transactions on Machine Learning Research*, 2023.

[Geilinger et al., 2020] Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. Add: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 39(6), 2020.

[Giftthaler et al., 2017] Markus Giftthaler, Michael Neunert, Markus Stäuble, Marco Frigerio, Claudio Semini, and Jonas Buchli. Automatic differentiation of rigid body dynamics for optimal control and estimation. *Advanced Robotics*, 31(22), 2017.

[Golemo et al., 2018] Florian Golemo, Adrien Ali Taiga, Aaron Courville, and Pierre-Yves Oudeyer. Sim-to-real transfer with neural-augmented robot simulation. In *Conference on Robot Learning*. PMLR, 2018.

[Goodfellow et al., 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[Grandia et al., 2023] Ruben Grandia, Farbod Farshidian, Espen Knoop, Christian Schumacher, Marco Hutter, and Moritz Bächer. DOC: Differentiable optimal control for retargeting motions onto legged robots. *ACM Trans. Graph.*, 42(4):1–14, July 2023.

[Grandia et al., 2024] Ruben Grandia, Espen Knoop, Michael A. Hopkins, Georg Wiedebach, Jared Bishop, Steven Pickles, David Müller, and Moritz Bächer. Design and Control of a Bipedal Robotic Character. In *Proceedings of Robotics: Science and Systems*, Delft, the Netherlands, July 2024.

[Guo et al., 2020] Chuan Guo, Xinxin Zuo, Sen Wang, Shihao Zou, Qingyao Sun, Annan Deng, Minglun Gong, and Li Cheng. Action2motion: Conditioned gen-

eration of 3d human motions. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2021–2029, 2020.

[Guo et al., 2022] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5142–5151, 2022.

[Hämäläinen et al., 2020] Perttu Hämäläinen, Juuso Toikka, Amin Babadi, and C Karen Liu. Visualizing movement control optimization landscapes. *IEEE Transactions on Visualization and Computer Graphics*, 28(3):1648–1660, 2020.

[Hartley et al., 2020] Ross Hartley, Maani Ghaffari, Ryan M Eustice, and Jessy W Grizzle. Contact-aided invariant extended kalman filtering for robot state estimation. *The International Journal of Robotics Research*, 39(4):402–430, 2020.

[Harvey et al., 2020] Félix G. Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. Robust motion in-betweening. *ACM Trans. Graph.*, 39(4), aug 2020.

[Hasenclever et al., 2020] Leonard Hasenclever, Fabio Pardo, Raia Hadsell, Nicolas Heess, and Josh Merel. CoMic: Complementary task learning & mimicry for reusable skills. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4105–4115. PMLR, 13–18 Jul 2020.

[He et al., 2022] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[Heess et al., 2017] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017.

[Heiden et al., 2021] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. In *IEEE ICRA*, 2021.

[Hendrycks and Gimpel, 2016] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[Heusel et al., 2017] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[Higgins et al., 2017] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.

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

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Holden et al., 2015] Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 technical briefs*, pages 1–4. 2015.

[Holden et al., 2016] Daniel Holden, Jun Saito, and Taku Komura. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)*, 35(4):1–11, 2016.

[Holden et al., 2017] Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Trans. Graph.*, 36(4), 2017.

[Hu et al., 2015] Liwen Hu, Chongyang Ma, Linjie Luo, and Hao Li. Single-view hair modeling using a hairstyle database. *ACM Trans. Graph.*, 34(4), jul 2015.

[Hu et al., 2020] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *ICLR*, 2020.

[Hwangbo et al., 2019] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[Jiang et al., 2024] Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. Motiongpt: Human motion as a foreign language. *Advances in Neural Information Processing Systems*, 36, 2024.

[Juravsky et al., 2022] Jordan Juravsky, Yunrong Guo, Sanja Fidler, and Xue Bin Peng. Padl: Language-directed physics-based character control. In *SIGGRAPH Asia 2022 Conference Papers*, New York, NY, USA, 2022. Association for Computing Machinery.

[Karunratanakul et al., 2023] Korrawe Karunratanakul, Konpat Preechakul, Supasorn Suwajanakorn, and Siyu Tang. Guided motion diffusion for controllable

human motion synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2151–2162, 2023.

[Karunratanakul et al., 2024] Korrawe Karunratanakul, Konpat Preechakul, Emre Aksan, Thabo Beeler, Supasorn Suwajanakorn, and Siyu Tang. Optimizing diffusion noise can serve as universal motion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1334–1345, 2024.

[Kazemnejad et al., 2023] Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[Ke et al., 2021] Guolin Ke, Di He, and Tie-Yan Liu. Rethinking positional encoding in language pre-training. In *International Conference on Learning Representations*, 2021.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kingma and Welling, 2013] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[Kloss et al., 2020] Alina Kloss, Stefan Schaal, and Jeannette Bohg. Combining learned and analytical models for predicting action effects from sensory data. *The International Journal of Robotics Research*, 2020.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[Langevin, 1908] Paul Langevin. Sur la théorie du mouvement brownien. *CR Acad. Sci. Paris*, 146(530-533):530, 1908.

[Lawrence and Lin, 1989] I Lawrence and Kuei Lin. A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, 1989.

[LeCun et al., 1989] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[LeCun et al., 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[Lee et al., 2010] Yoonsang Lee, Sungeun Kim, and Jehee Lee. Data-driven biped control. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, New York, NY, USA, 2010. Association for Computing Machinery.

[Lee et al., 2018] Kyungho Lee, Seyoung Lee, and Jehee Lee. Interactive character animation by learning multi-objective control. *ACM Transactions on Graphics (TOG)*, 37(6):1–10, 2018.

[Lee et al., 2023] Sunmin Lee, Taeho Kang, Jungnam Park, Jehee Lee, and Jungdam Won. Same: Skeleton-agnostic motion embedding for character animation. In *SIGGRAPH Asia 2023 Conference Papers*, New York, NY, USA, 2023. Association for Computing Machinery.

[Levine et al., 2012] Sergey Levine, Jack M. Wang, Alexis Haraux, Zoran Popović, and Vladlen Koltun. Continuous character control with low-dimensional embeddings. *ACM Trans. Graph.*, 31(4), jul 2012.

[Lim and Zohren, 2021] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philos. Transactions of the Royal Society A*, 2021.

[Lim et al., 2021] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4), 2021.

[Ling et al., 2020] Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel Van De Panne. Character controllers using motion vaes. *ACM Trans. Graph.*, 39(4), aug 2020.

[Liu and Hodgins, 2017] Libin Liu and Jessica Hodgins. Learning to schedule control fragments for Physics-Based characters using deep Q-Learning. *ACM Trans. Graph.*, 36(3):1–14, June 2017.

[Liu et al., 2020] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020.

[Liu et al., 2021] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[Liu et al., 2022] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[Loper et al., 2015] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[Loshchilov and Hutter, 2017] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

[Luo et al., 2020] Ying-Sheng Luo, Jonathan Hans Soeseno, Trista Pei-Chun Chen, and Wei-Chao Chen. CARL: controllable agent with reinforcement learning for quadruped locomotion. *ACM Trans. Graph.*, 39(4):38:1–38:10, August 2020.

[Luo et al., 2023] Zhengyi Luo, Jinkun Cao, Alexander W. Winkler, Kris Kitani, and Weipeng Xu. Perpetual humanoid control for real-time simulated avatars. In *International Conference on Computer Vision (ICCV)*, 2023.

[Luo et al., 2024] Zhengyi Luo, Jinkun Cao, Josh Merel, Alexander Winkler, Jing Huang, Kris M. Kitani, and Weipeng Xu. Universal humanoid motion representations for physics-based control. In *The Twelfth International Conference on Learning Representations*, 2024.

[Mahmood et al., 2019] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5442–5451, 2019.

[Makoviychuk et al., 2021] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[Merel et al., 2018] Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. Neural probabilistic motor primitives for humanoid control. In *International Conference on Learning Representations*, 2018.

[Merel et al., 2020] Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. Catch & carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Trans. Graph.*, 39(4):39:1–39:12, August 2020.

[Miki et al., 2022] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science robotics*, 7(62):eabk2822, 2022.

[Mildenhall et al., 2021] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65(1):99–106, dec 2021.

[Mixamo, 2024] Mixamo. Mixamo. animated 3d characters for games, film, and more., 2024.

[Nichol and Dhariwal, 2021] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 18–24 Jul 2021.

[Ouyang et al., 2022] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.

[Park et al., 2019a] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[Park et al., 2019b] Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. Learning predict-and-simulate policies from unorganized human motion data. *ACM Trans. Graph.*, 38(6):1–11, November 2019.

[Pascanu et al., 2013] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*. PMLR, 2013.

[Peebles and Xie, 2023] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4195–4205, October 2023.

[Peng et al., 2018a] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4), 2018.

[Peng et al., 2018b] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[Peng et al., 2021] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: adversarial motion priors for stylized physics-based

character control. *ACM Trans. Graph.*, 40(4), 2021.

[Peng et al., 2022] Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Trans. Graph.*, 41(4), 2022.

[Petrovich et al., 2021] Mathis Petrovich, Michael J. Black, and Gül Varol. Action-conditioned 3D human motion synthesis with transformer VAE. In *International Conference on Computer Vision (ICCV)*, 2021.

[Petrovich et al., 2022] Mathis Petrovich, Michael J. Black, and Gül Varol. TEMOS: Generating diverse human motions from textual descriptions. In *European Conference on Computer Vision (ECCV)*, 2022.

[Plappert et al., 2016] Matthias Plappert, Christian Mandery, and Tamim Asfour. The KIT motion-language dataset. *Big Data*, 4(4):236–252, dec 2016.

[Press et al., 2022] Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR*. OpenReview.net, 2022.

[Punnakkal et al., 2021] Abhinanda R. Punnakkal, Arjun Chandrasekaran, Nikos Athanasiou, Alejandra Quiros-Ramirez, and Michael J. Black. BABEL: Bodies, action and behavior with english labels. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 722–731, June 2021.

[Raab et al., 2023] Sigal Raab, Inbal Leibovitch, Peizhuo Li, Kfir Aberman, Olga Sorkine-Hornung, and Daniel Cohen-Or. Modi: Unconditional motion synthesis from diverse data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13873–13883, 2023.

[Raab et al., 2024] Sigal Raab, Inbal Leibovitch, Guy Tevet, Moab Arar, Amit H Bermano, and Daniel Cohen-Or. Single motion diffusion. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

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

[Radford et al., 2023] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings*

of Machine Learning Research, pages 28492–28518. PMLR, 23–29 Jul 2023.

[Raffel et al., 2020] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), jan 2020.

[Reallusion, 2024] Reallusion. 3d animation and 2d cartoons made simple., 2024.

[Rempe et al., 2021] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J. Guibas. Humor: 3d human motion model for robust pose estimation. In *International Conference on Computer Vision (ICCV)*, 2021.

[Ren et al., 2023] Jiawei Ren, Mingyuan Zhang, Cunjun Yu, Xiao Ma, Liang Pan, and Ziwei Liu. Insactor: Instruction-driven physics-based characters. *NeurIPS*, 2023.

[Rosu et al., 2022] Radu Alexandru Rosu, Shunsuke Saito, Ziyan Wang, Chenglei Wu, Sven Behnke, and Giljoo Nam. Neural strands: Learning hair geometry and appearance from multi-view images. *ECCV*, 2022.

[Rumelhart et al., 1986] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[Ruoss et al., 2023] Anian Ruoss, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, Róbert Csordás, Mehdi Bennani, Shane Legg, and Joel Veness. Randomized positional encodings boost length generalization of transformers. In *61st Annual Meeting of the Association for Computational Linguistics*, 2023.

[Schrittwieser et al., 2020] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[Schulman et al., 2016] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[Schulman et al., 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Schumacher et al., 2021] Christian Schumacher, Espen Knoop, and Moritz

Bächer. A versatile inverse kinematics formulation for retargeting motions onto robots with kinematic loops. *IEEE Robotics and Automation Letters*, 6(2):943–950, 2021.

[Serifi et al., 2023] Agon Serifi, Espen Knoop, Christian Schumacher, Naveen Kumar, Markus Gross, and Moritz Bächer. Transformer-based neural augmentation of robot simulation representations. *IEEE Robotics and Automation Letters*, 8(6):3748–3755, 2023.

[Serifi et al., 2024a] Agon Serifi, Ruben Grandia, Espen Knoop, Markus Gross, and Moritz Bächer. Robot motion diffusion model: Motion generation for robotic characters. In *ACM SIGGRAPH Asia 2024 Conference Papers*, SA '24, pages 1–9, 2024.

[Serifi et al., 2024b] Agon Serifi, Ruben Grandia, Espen Knoop, Markus Gross, and Moritz Bächer. Vmp: Versatile motion priors for robustly tracking motion on physical characters. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '24, pages 1–11, 2024.

[Shafir et al., 2023] Yoni Shafir, Guy Tevet, Roy Kapon, and Amit Haim Bermano. Human motion diffusion as a generative prior. In *The Twelfth International Conference on Learning Representations*, 2023.

[Smith, 2005] Russell Smith. Open dynamics engine. 2005.

[Sok et al., 2007] Kwang Won Sok, Manmyung Kim, and Jehee Lee. Simulating biped behaviors from human motion data. *ACM Trans. Graph.*, 26(3):107–es, jul 2007.

[Starke et al., 2019] Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. Neural state machine for character-scene interactions. *ACM Transactions on Graphics*, 38(6):178, 2019.

[Starke et al., 2020] Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. Local motion phases for learning multi-contact character movements. *ACM Trans. Graph.*, 39(4), aug 2020.

[Starke et al., 2022] Sebastian Starke, Ian Mason, and Taku Komura. DeepPhase: periodic autoencoders for learning motion phase manifolds. *ACM Trans. Graph.*, 41(4):1–13, July 2022.

[Su et al., 2023] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023.

[Suh et al., 2022] Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In *Inter-*

national Conference on Machine Learning, pages 20668–20696. PMLR, 2022.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Tancik et al., 2020] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[Team et al., 2025] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[Tessler et al., 2023] Chen Tessler, Yoni Kasten, Yunrong Guo, Shie Mannor, Gal Chechik, and Xue Bin Peng. Calm: Conditional adversarial latent models for directable virtual characters. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, New York, NY, USA, 2023. Association for Computing Machinery.

[Tessler et al., 2024] Chen Tessler, Yunrong Guo, Ofir Nabati, Gal Chechik, and Xue Bin Peng. Maskedmimic: Unified physics-based character control through masked motion inpainting. *ACM Transactions on Graphics (TOG)*, 43(6):1–21, 2024.

[Tevet et al., 2022] Guy Tevet, Brian Gordon, Amir Hertz, Amit H Bermano, and Daniel Cohen-Or. Motionclip: Exposing human motion generation to clip space. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 358–374. Springer, 2022.

[Tevet et al., 2023] Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. Human motion diffusion model. In *The Eleventh International Conference on Learning Representations*, 2023.

[Tevet et al., 2025] Guy Tevet, Sigal Raab, Setareh Cohan, Daniele Reda, Zhengyi Luo, Xue Bin Peng, Amit Haim Bermano, and Michiel van de Panne. CLoSD: Closing the loop between simulation and diffusion for multi-task character control. In *The Thirteenth International Conference on Learning Representations*, 2025.

[Todorov et al., 2012] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012.

[Torres et al., 2021] José F Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez, and Alicia Troncoso. Deep learning for time series forecast-

ing: a survey. *Big Data*, 9(1), 2021.

[Touvron et al., 2021] Hugo Touvron, Matthieu Cord, Alaaeldin El-Nouby, Piotr Bojanowski, Armand Joulin, Gabriel Synnaeve, and Hervé Jégou. Augmenting convolutional networks with attention-based aggregation. *CoRR*, abs/2112.13692, 2021.

[Tunyasuvunakool et al., 2020] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

[Vaswani et al., 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017.

[Wampler et al., 2014] Kevin Wampler, Zoran Popović, and Jovan Popović. Generalizing locomotion style to new animals with inverse optimal regression. *ACM Transactions on Graphics (TOG)*, 33(4):1–11, 2014.

[Wang et al., 2020] Tingwu Wang, Yunrong Guo, Maria Shugrina, and Sanja Fidler. Unicon: Universal neural controller for physics-based character motion, 2020.

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[Williams and Zipser, 1989] Ronald Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1989.

[Williams, 1992] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[Won and Lee, 2019] Jungdam Won and Jehee Lee. Learning body shape variation in physics-based characters. *ACM Trans. Graph.*, 38(6), nov 2019.

[Won et al., 2020] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Trans. Graph.*, 39(4):33:1–33:12, August 2020.

[Won et al., 2022] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. Physics-based character controllers using conditional VAEs. *ACM Trans. Graph.*, 41(4):1–12, July 2022.

[Xie et al., 2020] Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. Allsteps: Curriculum-driven learning of stepping stone skills.

Computer Graphics Forum, 39(8):213–224, 2020.

[Xie et al., 2024] Yiming Xie, Varun Jampani, Lei Zhong, Deqing Sun, and Huaizu Jiang. Omnicontrol: Control any joint at any time for human motion generation. In *The Twelfth International Conference on Learning Representations*, 2024.

[Xu et al., 2023] Pei Xu, Xiumin Shang, Victor Zordan, and Ioannis Karamouzas. Composite motion learning with task control. *ACM Transactions on Graphics*, 42(4), 2023.

[Yao et al., 2022] Heyuan Yao, Zhenhua Song, Baoquan Chen, and Libin Liu. ControlVAE: Model-Based learning of generative controllers for Physics-Based characters. *ACM Trans. Graph.*, 41(6):1–16, November 2022.

[Yao et al., 2024] Heyuan Yao, Zhenhua Song, Yuyang Zhou, Tenglong Ao, Baoquan Chen, and Libin Liu. Moconvq: Unified physics-based motion control via scalable discrete representations. *ACM Trans. Graph.*, 43(4), jul 2024.

[Yuan et al., 2023] Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. Physdiff: Physics-guided human motion diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[Zhang et al., 2021] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3), 2021.

[Zhang et al., 2023] Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Shaoli Huang, Yong Zhang, Hongwei Zhao, Hongtao Lu, and Xi Shen. T2m-gpt: Generating human motion from textual descriptions with discrete representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[Zhang et al., 2024] Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[Zhao et al., 2020] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pages 737–744. IEEE, 2020.

[Zhou et al., 2019] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019.

[Zhou et al., 2023] Yuxiao Zhou, Menglei Chai, Alessandro Pepe, Markus Gross, and Thabo Beeler. Groomgen: A high-quality generative hair model using hierarchical latent representations. *ACM Transactions on Graphics*, 42(6):1–16, December 2023.

[Zhu et al., 2023] Qingxu Zhu, He Zhang, Mengting Lan, and Lei Han. Neural categorical priors for physics-based character control. *ACM Trans. Graph.*, 42(6), dec 2023.

Declaration of the use of AI-based tools

The following Generative AI tools have been used for advanced writing assistance.

Tool	Use Case	Comments
GPT-4o[1]	Revision of text / German translation	Prompts similar to "List synonyms of [word] for: [sentence]", "Improve: [sentence]", and grammatical or structural questions like "is this grammatically correct? [text]", "What tense to use in acknowledgments?". No text was ever blindly copied or generated from scratch.
Grammarly[2]	Grammar/Punctuation/ Spelling	To check and fix grammar, punctuation, and spelling.

[1] `https://chatgpt.com/` (last date of access: 15.05.2025)
[2] `https://app.grammarly.com/` (last date of access: 15.05.2025)