UNIVERSITY OF CALIFORNIA

Los Angeles

Learning-Based, Muscle-Actuated Biomechanical Human Animation: Bipedal Locomotion Control and Facial Expression Transfer

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Wuyue Lu

2024

ABSTRACT OF THE DISSERTATION

Learning-Based, Muscle-Actuated Biomechanical Human Animation:
Bipedal Locomotion Control and Facial Expression Transfer

by

Wuyue Lu
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2024
Professor Demetri Terzopoulos, Chair

This dissertation explores the frontier of biomimetic virtual human animation. Applying state-of-the-art machine learning techniques, we develop neuromuscular control frameworks that significantly enhance the naturalness and realism of simulated human motions. In particular, we address the challenges of achieving locomotion-based animation with biomechanical musculoskeletal fidelity and in the transfer of head pose and facial expressions from images and videos to a muscle-actuated model of the face-head-neck biomechanical complex. Key technical contributions of the thesis include (1) integration of Central Pattern Generator (CPG) controllers with a biomechanical body model, which enables the generation of adaptive and flexible locomotion patterns through reinforcement learning-based control parameter optimization, and (2) leveraging of the Facial Action Coding System in a computer-vision-based estimation of facial expressions and their transfer to a 3D face model via the coordinated activation of the muscles of facial expression. With regard to implementation, the migration of our simulation environments to Nvidia's GPU-accelerated Omniverse platform affords improved computational performance and advanced rendering techniques. By demonstrating the versatility of machine learning applied to muscle-driven face and body animation, this work advances the exploitation of biomimetic, physics-based human modeling and simulation in computer graphics and vision.

The dissertation of Wuyue Lu is approved.

Bolei Zhou

Chenfanfu Jiang

Joseph Distefano III

Demetri Terzopoulos, Committee Chair

University of California, Los Angeles

2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Firstly, I would like to express my profound gratitude to Professor Demetri Terzopoulos for his support and guidance throughout my PhD program at UCLA. In the summer of 2018, I participated in the Cross-disciplinary Scholars in Science and Technology (CSST) program at UCLA, marking the first time I collaborated with Professor Terzopoulos. Through that experience, I discovered my passion for research in computer graphics and vision, which has reinforced my dedication to this field. I subsequently joined the UCLA Computer Graphics & Vision Lab the following year, embarking on a transformative five-year journey that has culminated in this dissertation. During my PhD, Professor Terzopoulos provided invaluable insights, encouragement, and mentorship, not only in research but also in various aspects of life. His unwavering commitment to excellence and his ability to balance innovation with practicality have been a constant source of inspiration for me. I am very grateful for the opportunity to benefit from his wealth of knowledge and experience.

Secondly, I am incredibly grateful to my dissertation committee members, Professors Bolei Zhou, Chenfanfu Jiang, and Joseph Distefano III, for generously dedicating their time to serve on my committee and providing invaluable guidance that has helped shape my research. Their expertise and unique perspectives have enriched my work, encouraging me to think more critically about my findings and supporting the successful completion of my Ph.D.

I would also like to extend my appreciation to Dr. Tao Zhou and Dr. Masaki Nakada, our esteemed former colleagues, who have continued to provide invaluable assistance and support to our lab even after their graduation. Their willingness to share their expertise and offer helpful suggestions on improving established projects and codebases has been crucial in advancing our research. Moreover, their insights on future directions have inspired us and helped shape the trajectory of our work. I am truly grateful for their ongoing dedication and significant impact on my research journey.

Moreover, appreciation should also go to the collaborators who have significantly contributed to the two main projects presented in this dissertation. Surya Dwarakanath and Dr. Xiao Zeng have been invaluable co-authors of our facial expression transfer project, bringing their expertise and dedication to the research. Their insights and efforts have greatly enhanced the quality and impact of our work. Additionally, I am grateful for the assistance provided by Avalon Vinella and Haotian Yi, who generously volunteered their time and skills to support our research endeavors. Their contributions have been instrumental in advancing our projects and achieving our research goals. I am truly thankful for the opportunity to work alongside such talented and committed individuals.

My sincere thanks also go to the excellent staff in the UCLA Computer Science Department's Graduate Student Affairs Office, particularly Joseph Brown, Juliana Alvarez, and Helen Tran. Their tireless efforts and support, especially during the challenging times of the pandemic, have been invaluable to my success as a graduate student.

Last but not least, I would like to express my heartfelt gratitude to my family and friends for their unwavering support, patience, and understanding throughout this challenging journey. Their constant encouragement and belief in me have been a driving force, motivating me to persevere through the most difficult times. I am especially indebted to my parents, whose unconditional love, sacrifices, and guidance have shaped me into the person I am today. Without their support, this achievement would not have been possible.

VITA

2015 – 2019	B.S. in Computer Science University of Science and Technology Hefei, China
2019 – 2022	M.S. in Computer Science University of California, Los Angeles Los Angeles, CA
2022	Software Engineer Intern Apple, Inc. Cupertino, CA
2023	Software Engineer Intern Google, Inc. Mountain View,, CA
2020 – 2023	Teaching Assistant Computer Science Department University of California, Los Angeles Los Angeles, CA
2019 – 2020	Graduate Student Researcher Computer Science Department University of California, Los Angeles Los Angeles, CA
2022 – 2024	PhD Candidate Computer Science Department University of California, Los Angeles Los Angeles, CA

PUBLICATIONS

Hatamizadeh, A., Hoogi, A., Sengupta, D., Lu, W., Wilcox, B., Rubin, D., & Terzopoulos, D., "Deep active lesion segmentation," in *Machine Learning in Medical Imaging: 10th*

International Workshop, MLMI 2019, Proceedings 10, pp. 98–105, 2019. (Hatamizadeh et al., 2019)

Lu, W., & Liu, L., “Surface reconstruction via cooperative evolutions,” *Computer Aided Geometric Design*, Vol. 77, p. 101831, 2020. (Lu and Liu, 2020)

Zeng, X. S., Dwarakanath, S., Lu, W., Nakada, M., & Terzopoulos, D., “Facial expression transfer from video via deep learning,” in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 1–2, 2021. (Zeng et al., 2021a)

Zeng, X. S., Dwarakanath, S., Lu, W., Nakada, M., and Terzopoulos, D., “Neuromuscular control of the face-head-neck biomechanical complex with learning-based expression transfer from images and videos,” in *Advances in Visual Computing: 16th International Symposium, ISVC 2021, Virtual Event, October 4–6, 2021, Proceedings, Part I*, pp. 116–127, 2021. (Zeng et al., 2021b)

CHAPTER 1

Introduction

Modern computer graphics technologies have yielded graphical characters with increasingly realistic appearance and human resemblance. Enormous efforts have been devoted in the pursuit of photo-realistic graphics and, with the support of modern graphic devices and rendering techniques, this has been successful for static scenes. Especially in the case of human animation, however, the audience has become increasingly sensitive to the annoying unpleasantness of even minor imperfections, which researchers have described as the "Uncanny Valley" phenomenon (Mori et al., 2012) (Figure 1.1).

Consequently, the increasing demand from the modern motion picture and game industries for visually natural, high-fidelity human characters animated with physically

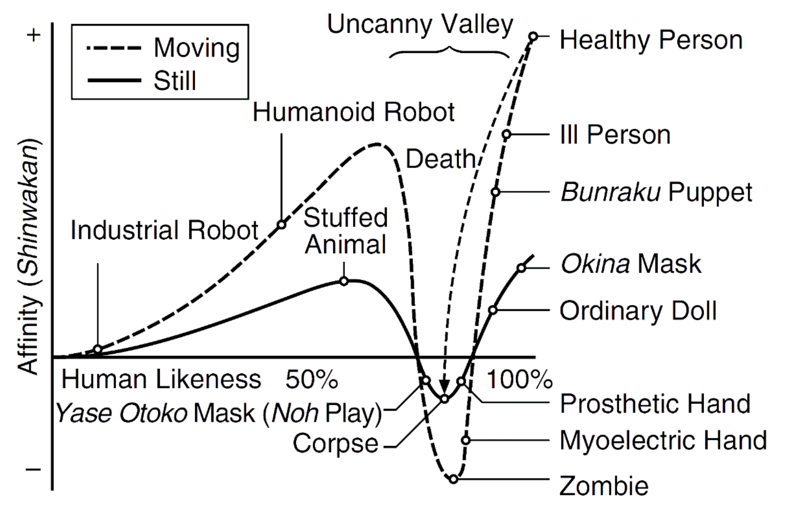


Figure 1.1: The "Uncanny Valley" phenomenon, illustrating the relationship between an object's human likeness and its perceived familiarity or affinity. The graph shows that as human likeness increases, the perceived affinity increases until a point where it sharply drops. This valley-like dip is more pronounced for moving objects compared to still objects. Adapted from (Mori et al., 2012).

realistic motions has motivated considerable recent research into biomechanical human modeling and simulation. Meanwhile, massive improvements in computational power through Graphics Processing Units (GPUs) has enabled the efficient simulation of anatomically accurate, full-body human models with heightened complexity and biomechanical fidelity, although a trade-off between accuracy and real-time performance is still necessary for most applications.

This dissertation presents innovative techniques for synthesizing animations of the human face and body through biomechanical models. Motion capture techniques have long been the industry standard in human character animation. While offering high motion fidelity and realism, motion capture has tended to suffer from high equipment costs and technical complexity (Kitagawa and Windsor, 2020). Furthermore, editing and adapting pre-recorded motion capture data into dynamic scenarios presents significant challenges (Salisu et al., 2023; Wade et al., 2022). By contrast, animation through physics-based simulation can, in principle, provide a wide range of motion on the fly, adapting to rich scenarios and unique physical constraints. Such adaptability is critical in dynamic environments such as interactive games or virtual reality, where characters must respond to unpredictable elements and user input in a lifelike manner. However, replicating human body movements and facial expressions with biomechanical fidelity, down to individual skeletal and facial muscle activations, presents significant challenges.

1.1 Scope of the Research

In the pursuit of more natural and realistic character animation, this thesis explores a fusion of machine learning with muscle-actuated biomechanical simulation and animation techniques. The main research topics that we pursue are as follows:

- **Musculoskeletal Body Control:** In the field of full-body musculoskeletal model control, the objective is to simulate human anatomy and movement with biomechanical fidelity. Research in this area is fundamental not only for creating realistic digital characters or artificial life, but also for advancing prosthetic design and rehabilitation techniques for impaired individuals. A critical trade-off of this topic is the balance between simulation accuracy and computational efficiency. Despite the advances in computing power, the complexity of the human body still poses significant challenges to parallel computing. This complexity encourages the increasing integration of state-of-the-art machine learning techniques into this research topic; e.g., deep learning and reinforcement learning.

- **Central Pattern Generators:** CPGs are self-organized biological neural circuits that are fundamental to the control of rhythmic, repetitive motor behaviors, such as walking, running, and swimming. They are helpful in the biomimetic modeling of vertebrates, including humans, because of their ability to autonomously generate periodic signals. Research in this area has two main focuses: the biomimetic modeling and implementation of CPGs in simulated characters; and fine-tuning CPG parameters to achieve precise motion control. Traditionally, CPG parameter tuning has been manual or based on genetic algorithms. However, recent advances have shifted the focus to machine learning techniques, particularly reinforcement learning, to optimize these parameters more efficiently and effectively.

- **The Face-Head-Neck Complex and Facial Expression:** Biomechanical modeling with a focus on the neuromuscular control of the face-head-neck complex, is a cross-disciplinary domain that combines insights from neuroscience, biomechanics, and computer graphics. It involves understanding the interactions between the muscles and neurons that are responsible for human facial expression. Research into these connections is essential for achieving consistent expression representation across different simulation environments and scenarios, thus contributing to the development of digital human characters. Meanwhile, the cross-disciplinary nature of this field, which requires synthesizing diverse forms of information, highlights the significant role of machine learning in effectively bridging these gaps.

- **Learning-Based Expression Transfer:** Transferring facial expressions from images or videos to face models is a classic challenge in computer graphics. It often involves studying a generalized parametric representation of human facial expressions and adapting this intermediate representation to 3D face models through mesh deformation or shape blending. A more biomimetic and natural approach, however, is to transfer expressions to anatomically accurate musculoskeletal face models. In this context, the Facial Action Coding System (FACS) Ekman and Friesen (1978); Cohn et al. (2007) proves to be an effective intermediate representation, which encodes expressions into distinct action units (AUs), each of which is correlated with specific coordinated facial muscle activations, thus making it particularly suitable for expression transfer through facial muscle activation and control.

In short, the scope of the thesis includes biomechanical, muscle-actuated bipedal human locomotion and facial animation. For the former, we integrate reinforcement learning into a Central Pattern Generator (CPG) skeletal muscle-control structure, thus circumventing the need for laborious manual fine-tuning of CPG neurons in traditional methods. For the latter, we utilize the FACS as an intermediate representation, and deep learning techniques to effectively transfer facial expressions from images and video into facial muscle activations that animate a model of the face-head-neck musculoskeletal complex.

1.2 Contributions

This thesis combines research projects that share a common theme of biomimetic virtual human modeling and control. Based on earlier PhD thesis research conducted in the UCLA Computer Graphics & Vision Laboratory (Lee et al., 2009; Nakada et al., 2018), which first developed a biomechanical model of the face-head-neck complex and, ultimately, a high-fidelity, full-body biomechanical human musculoskeletal model, our present work makes several key contributions:

1. We migrate our high-fidelity biomechanical human model to Nvidia's Omniverse platform. The model employed advanced hybrid system dynamics algorithms Lee

et al. (2009), but was constrained by its limited GPU parallelism. We demonstrate the improved performance enabled by PhysX, Omniverse's GPU-friendly multi-body physics simulation system. We also present new simulation capabilities, such as soft-body dynamics and simulation. Additionally, the platform's cutting-edge ray tracing integration significantly improves our simulations' graphical performance and visual realism.

2. We integrate the CPG structure into the model's neuromuscular control system. We improve the existing CPG implementation designed for robotics and locomotion studies and make its parametric representation more suitable for machine learning-based optimization. By utilizing this biomimetic structure that intrinsically exerts rhythmic control signals, we establish a foundation for generating natural and adaptive locomotion patterns.

3. We explore the use of reinforcement learning techniques to optimize the control parameters of the CPG-based locomotion system. While pure CPG control signals can effectively actuate motions such as salamander crawling (Ijspeert et al., 2007), human bipedal motions require additional considerations to maintain balance and respond to external perturbations. To address this challenge, we propose a novel architecture that cohesively tunes CPG and reflex control signals. In addition, we propose a reward design that encourages agents to mimic reference motions while maintaining natural stepping behaviors. This approach combines the strengths of both biologically inspired CPG structures and feedback-based reflex control.

4. We introduce a novel framework for animating human facial expressions and head movements using our biomechanical face-head-neck model Lee and Terzopoulos (2006). Our framework features a controller that efficiently translates human facial expressions into the intermediate representation of FACS AUs. It then maps these to the activations of muscle actuators that drive the biomechanical model. Our approach is versatile and can be adapted to any muscle-driven facial model.

1.3 Thesis Overview

The subsequent chapters of this dissertation are as follows:

In Chapter 2, we review related work published in the existing literature.

In Chapter 3, we introduce our biomechanical human musculoskeletal model and explain the underlying control systems.

In Chapter 4, we present the design and implementation of the Central Pattern Generator for human locomotion control, accompanied by extensive experiments and discussions.

In Chapter 5, we discuss our work on deep learning-based human facial expression transfer, emphasizing the biomechanical plausibility of our approach.

In Chapter 6, we present our conclusions and discuss promising avenues for future work.

CHAPTER 2

Related Work

Our work represents a cross-disciplinary research endeavor that integrates insights from multiple fields, including biomechanical modeling, neuromuscular control, facial animation, and machine learning. In this chapter, we examine the relevant techniques and advancements in each of these areas, emphasizing their relevance to our research and explaining the major challenges researchers face.

2.1 Biomechanical Human Modeling and Simulation

Biomechanical modeling and simulation of the human body have gained significant attention in the field of computer graphics, with researchers focusing on creating anatomically accurate representations of the human body and its motions. The musculoskeletal system (Nordin, 2020), which accounts for the dynamics of bones and muscles, has been a primary target of human modeling that is relevant to this dissertation. Such models have evolved from early efforts in modeling specific body parts, such as the hand (Sueda et al., 2008; Van Nierop et al., 2008; Tsang et al., 2005), torso (DiLorenzo et al., 2008; DiLorenzo, 2009), face (Sifakis et al., 2005; Kähler et al., 2002; Lee et al., 1995), and neck(Lee and Terzopoulos, 2006), to more comprehensive full-body models (Rajagopal et al., 2016; Si et al., 2014; Nakada et al., 2018; Lee et al., 2019; Zhou, 2019).

State-of-the-art biomechanical models have achieved remarkable complexity and anatomical fidelity, enabling the simulation of a wide range of human movements and features, from subtle facial expression control (Lee and Terzopoulos, 2006) to complex motions that involve the whole body (Song et al., 2021). For example, Si et al. (2014) developed a comprehensive full-body musculoskeletal model incorporating more than 800 muscle actuators and dynamic soft tissue simulation that exhibited realistic locomotion-based swimming patterns in a simulated fluid environment, and Lee et al. (2019) implemented a full-body musculoskeletal model comprising 346 muscles that produced realistic human motions, including walking, running, and jumping.

These models often utilize advanced physics simulation techniques, such as Featherstone's algorithm (Featherstone, 2014) and Rosenbrock methods (Van Den Bogert et al., 2011), to compute the dynamics of the musculoskeletal system efficiently. However, the increasing complexity of these models poses challenges in achieving real-time performance (Phinyomark et al., 2018) or generating large amounts of data for machine-learning purposes. Our research constantly faces the dilemma of biomechanical fidelity and sample efficiency, and the main chapters in this thesis will explain our strategy to balance them under different circumstances.

2.2 Learning-Based Neuromuscular Control

Neuromuscular control of biomechanical models uses biomimetic approaches to replicate the neurofeedback and muscular responses observed in the human body. Pioneering in this field, the early work by Lee and Terzopoulos (2006) utilized neuromuscular learning methods to control a biomechanical model of the human neck-head complex driven by a total of 72 Hill-type muscles. They employed traditional artificial neural networks trained offline through backpropagation to map target head orientations to muscle activations required to achieve those orientations in physical simulation subject to gravity and other external forces. While this conventional neural network design, consisting of only a few hidden layers, can effectively handle a limited number of dimensions, it struggles to cope with higher dimensionalities. Their subsequent work (Lee et al., 2009) extended this control scheme to the entire upper human body but still faced challenges due to the limitations of machine learning at the time.

Over the recent decades, the field of machine learning has seen substantial advancements. Deep learning techniques, such as deep reinforcement learning (DRL), convolutional neural networks (CNNs), and recurrent neural networks (RNNs), have proven effective in handling high-dimensional data and capturing complex patterns (LeCun et al., 2015). Researchers have explored these new learning architectures and approaches to neuromuscular control (Nakada et al., 2018; Zhou, 2019). Peng et al. (2018) and Lee et al. (2019) utilized deep reinforcement learning to train policies that generate intermediate representations of desired local poses. Furthermore, Nakada et al. (2018) employed vision-based CNN models for biomimetic eye control, demonstrating the potential of deep learning to integrate sensory information for sensorimotor control.

Deep learning models are often considered "black boxes" due to their complex architectures and large numbers of trainable parameters. This lack of interpretability can impede the trust and adoption of these models in critical applications, such as medical analysis and rehabilitation (Tjoa and Guan, 2020; Holzinger et al., 2019). One approach to overcome this drawback is to incorporate domain knowledge into the architecture. For instance, Jiang et al. (2019); Peng and Van De Panne (2017) integrated physiological constraints and state-dependent torque limits into the deep reinforcement learning to improve naturalness. In a similar manner, our approach incorporates a biomimetic structure, Central Pattern Generators (CPGs), into our learning procedure, adding intrinsic naturalness and biomechanical realism to the learned movements.

2.3 Central Pattern Generator

Central Pattern Generators (CPGs) are neural networks capable of producing rhythmic patterns without sensory feedback, making them particularly useful for simulating and controlling intrinsic rhythmic movements, such as walking or swimming, in biomechanical models (Ijspeert, 2008). CPGs have been widely used to model locomotion in lower-level organisms, such as reptiles, and the basic locomotive patterns in higher-level ones, such as mammals. For example, Ijspeert et al. (2007) presented a salamander robot that was driven by a CPG-based spinal cord model together with PD controllers.

In the context of human locomotion control, CPGs have been employed to generate stable and natural walking patterns for bipedal models. Taga (1995) built a musculoskeletal model of eight links and twenty muscles, which were simulated in two dimensions and driven by seven pairs of neural oscillators. This model demonstrated the emergence of basic walking gaits with muscle force patterns aligned with measured EMG reference. Additionally, the human swimming locomotion model developed by (Si et al., 2014) also utilized the CPGs but in a different manner, where the generated cyclic signals represent the desired muscle lengths over time, and the muscle activations are calculated in accordance with the biomechanical Hill-type muscle modeling. However, the integration of CPGs with high-dimensional, muscle-actuated biomechanical models remains a significant challenge due to the inherent complexity of the neuromuscular control problem (Dzeladini et al., 2018).

Our research explores the integration of CPGs with deep reinforcement learning techniques to enable the control of complex biomechanical models for human locomotion. By leveraging the intrinsic naturalness brought by CPGs, we aim to develop a flexible and robust framework for generating stable and adaptable walking patterns in high-dimensional, muscle-actuated models.

2.4 Facial Animation and Expression Transfer

Facial animation has been a long-standing challenge in computer graphics. While techniques such as mesh deformation and texture animation are capable of animating simple facial expressions (Parke, 1972; Waters, 1987), incorporating physics simulation and neuromuscular control into the face model enhances anatomical realism and transferability. Pioneering works, such as those of Sifakis et al. (2005) and Lee and Terzopoulos (2006), constructed anatomically accurate face models driven by muscle activations. However, their systems require extensive manual setup and parameter tuning due to the complexity of facial soft-tissue structures. Moreover, such methods often rely on reference motion capture data from specialized devices, which can be a limitation.

To address these challenges, recent work in the field has explored data-driven approaches. For instance, Weise et al. (2011) and Li et al. (2013) introduced a facial performance capture framework based on an adaptive PCA model, which achieves real-time performance and is free of manual calibration. Similarly, Edwards et al. (2016) developed JALI, which achieves expressive lip synchronization through an animator-centric viseme model. Many researchers also utilized the Facial Action Coding System (FACS) (Ekman and Friesen, 1978) as an intermediate representation, encoding facial expressions and head poses as a vector of Action Units (AUs).

Our research focuses on developing a deep learning-based approach for facial expression transfer that can map facial expressions from 2D images or videos to a 3D biomechanical model of the face via neuromuscular control. Unlike many recent works following an end-to-end design trend, we incorporate domain knowledge to ensure biomimetic naturalness.

2.5 Deep Reinforcement Learning in Character Animation

Deep reinforcement learning (DRL) has proven to be a powerful tool for learning complex control policies in high-dimensional environments (Schulman et al., 2017). In the context of character animation, researchers have applied DRL to learn locomotion skills for both bipedal and quadrupedal characters (Peng et al., 2016, 2017), enabling characters to interact with complex environments. For instance, Peng et al. (2018) introduced DeepMimic, a DRL-based framework that leverages reference motion data to guide the learning process. Moreover, Merel et al. (2019) discussed the hierarchical motor control in mammal models, arguing that biomimetic hierarchical structures benefit the learning of complex and sophisticated motor skills.

However, the application of DRL to muscle-actuated biomechanical models remains a challenge due to the high dimensionality of the action space and the difficulty in defining suitable reward functions. Lee et al. (2019) addressed this challenge by proposing a scalable, two-level imitation learning algorithm. In their approach, an upper-level trajectory mimicking module outputs the desired joint angles, while a lower-level muscle coordination module determines the muscle activations required to achieve them.

Our research investigates the integration of DRL with CPGs and neuromuscular control to learn stable and adaptable locomotion controllers for high-dimensional, muscle-actuated biomechanical models, leveraging the strengths of hierarchical learning and biomimetic structures.

CHAPTER 3

Biomechanical Simulation and Control

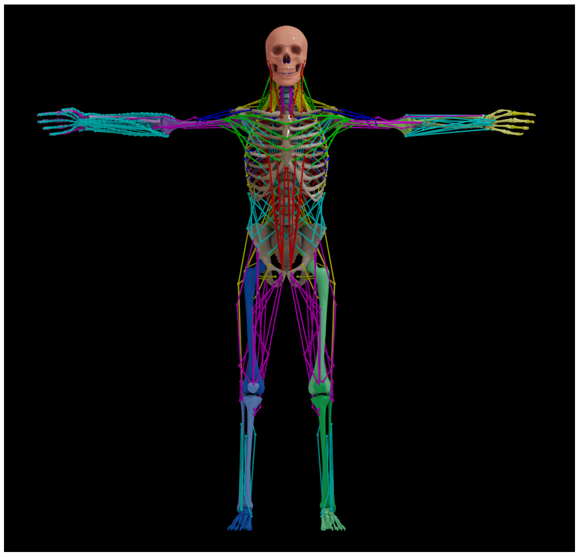
This chapter covers the simulation fundamentals of our virtual human body model, including musculoskeletal simulation for full-body models and the soft-tissue simulation techniques for facial expressions.

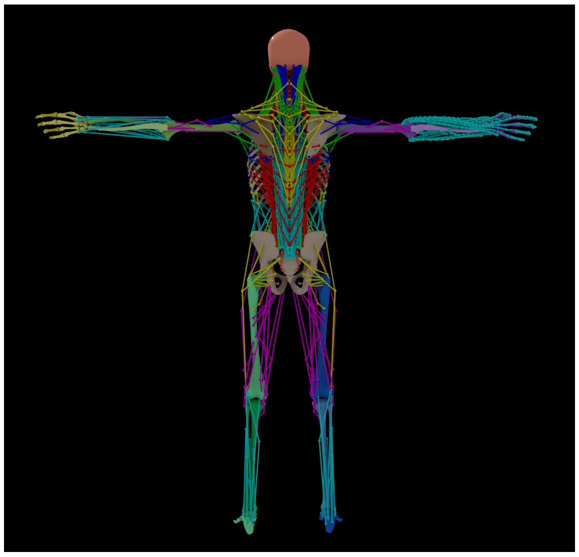
3.1 Overview

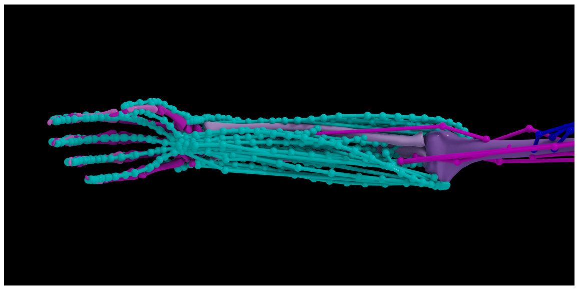
We build our biomechanical human model upon two primary layers for physics simulation: an articulated rigid-body system representing the skeleton and a refined version of the Hill-type muscle system (Zajac, 1989) for actuation. Together, these components form an anatomically accurate musculoskeletal simulator. The geometric data for our model are derived from *The Ultimate Human* model, a comprehensive commercial 3D human anatomy model designed for educational and medical purposes, ensuring high anatomical fidelity and realism.

In addition to the skeletal and muscular systems, our model also simulates soft tissue and skin, adding another layer of realism. The geometry of these soft tissues is influenced by, and in turn influences, the underlying skeletal and muscular systems. Accordingly, this thesis focuses on applying this layer to the simulation and learning of facial expressions.

Figure 3.1 shows the musculoskeletal simulation of the latest full-body model with additional complexity on the right hand, modeled and rendered in the Nvidia Omniverse framework. We can modify the complexity of each body part according to different purposes and applications. The majority of our experiments are conducted using the full-complexity model without the extended muscle actuators on the hands.


(a) Frontal view of full-body


(b) Dorsal view of full-body


(c) Frontal view of right hand

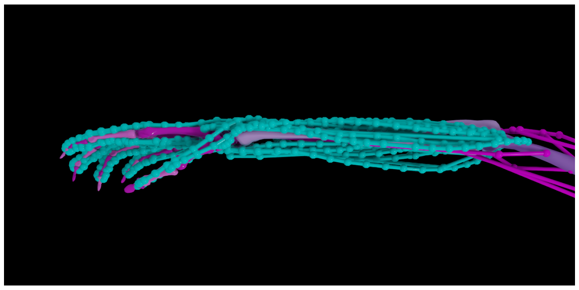
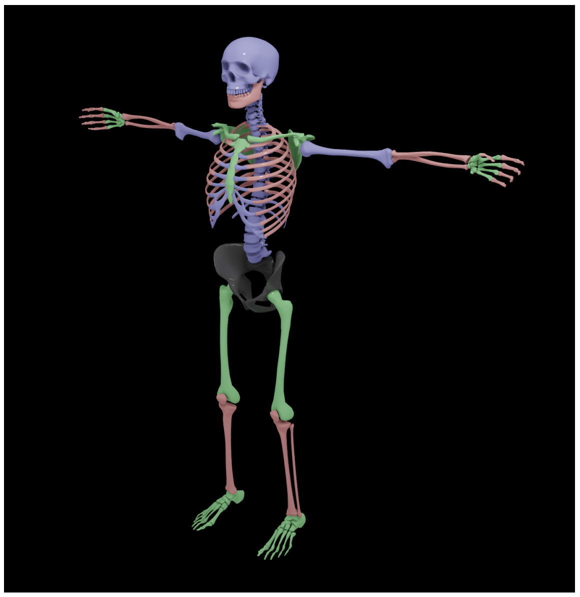

(d) Top-down view of right hand

Figure 3.1: Comprehensive anatomical renderings of our virtual human model's skeletal and muscular systems. Frontal (a) and dorsal (b) views are color-coded to delineate muscle groups. Close-up views of the right hand (c, d) detail the additional complexity of the musculoskeletal structure.

In our simulation, the skeletal system not only shapes the model's rendering and physical properties but also interfaces dynamically with the muscle system and soft tissue. Control agents, such as reflex controllers and Central Pattern Generators (CPGs), dynamically adjust muscle activations in response to sensory feedback and temporal signals. Subsequent sections will explore these interactive layers in depth, extensively referencing existing literature (Lee et al., 2009; Lee and Terzopoulos, 2006) and our investigations on facial models (Zeng et al., 2021a,b).


(a) Frontal view

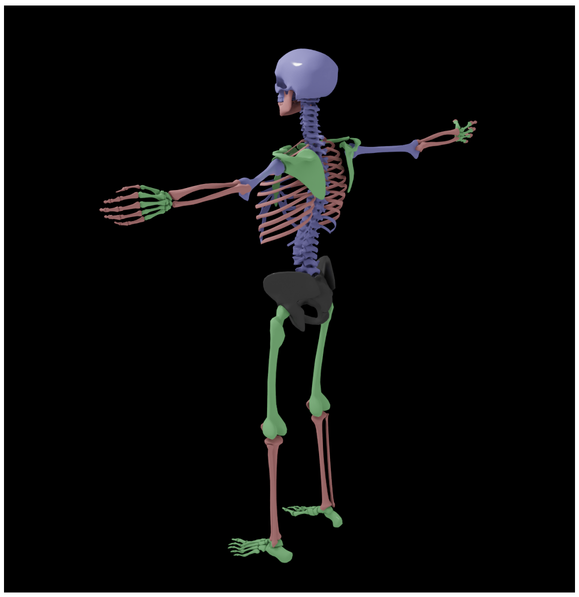

(b) Dorsal view

Figure 3.2: The bones are colored according to the DOF types of their connected joints. Ball joints (3 rotational DOFs) are colored blue; universal joints (2 rotational DOFs) are colored green; and revolute joints (1 rotational DOF) are colored red.

3.2 Musculoskeletal Simulation

3.2.1 Skeletal System

The biomechanical human model's skeleton is represented by an articulated multi-body system consisting of 103 rigid bone links, each of which has unique physical properties. These bones are configured with mass, inertia, and joint kinematic constraints derived from the aforementioned anatomical reference and geometrical model data. The skeletal system includes joints with appropriate rotational degrees of freedom (DOFs). For example, the cervical spine consists of seven stacked vertebrae that have 3-DOF rotational motion at the joints, while the ribs and costal cartilages have more constraints. Figure 3.2 categorizes the bones by their joint types using a color scheme.

The equations of motion of the skeleton system are written as

$$\mathbf{M}(\boldsymbol{q}) \begin{bmatrix} \ddot{\boldsymbol{q}}_m \\ \ddot{\boldsymbol{q}}_p \end{bmatrix} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \mathbf{P}(\boldsymbol{q}) \begin{bmatrix} \boldsymbol{f}_c \\ \mathbf{0} \end{bmatrix} + \mathbf{J}^T \boldsymbol{f}_e, \tag{3.1}$$

where $\boldsymbol{q}_m$ and $\boldsymbol{q}_e$ refer to the generalized coordinates of the joints that are torque-driven and passive, respectively, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are the joint velocities and accelerations, and $\mathbf{M}(\boldsymbol{q})$ is the mass matrix, $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ accounts for the Coriolis forces and centrifugal forces from the motion, together with the force from connecting tissues and muscle parallel elements ($\boldsymbol{f}_p$). On the right-hand side of the above equation, $\boldsymbol{f}_c$ represents the contractile muscle force generated by muscles with efferent activation inputs, and $\mathbf{P}(\boldsymbol{q})$ is the moment arm matrix that maps these muscle forces to joint torques. The computation technique for the moment arm matrix is introduced in (Gonzalez et al., 1997). Finally, $\boldsymbol{f}_e$ is the external forces, and $\mathbf{J}$ is the Jacobian matrix that maps them to torques.

Physically, the first term on the right-hand side of Equation 3.1 encapsulates the total force generated by the internal muscular system. The second term accounts for external influences such as gravity, support from surfaces, and interactions with objects or other environmental factors. By shifting the Coriolis force component $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ to the right and rearranging terms, we streamline the equations as follows:

$$\ddot{\boldsymbol{q}} = \phi(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}), \tag{3.2}$$

where $\boldsymbol{\tau}$ includes the joint torques resulting from muscle and external forces.

Following the approach proposed by (Lee et al., 2009), forward dynamics is used to compute ϕ by calculating $\ddot{\boldsymbol{q}}$ from the generated joint torques. We then apply the implicit Euler time integration method to determine the velocity term $\dot{\boldsymbol{q}}(t + \Delta t)$ at the next time step, which involves solving:

$$\dot{\boldsymbol{q}}(t + \Delta t) - \dot{\boldsymbol{q}}(t) = \Delta t \phi \left(\boldsymbol{q}(t + \Delta t), \dot{\boldsymbol{q}}(t + \Delta t), \boldsymbol{\tau}\right). \tag{3.3}$$

Despite the presence of variables at the next time step on the right-hand side, the problem may be simplified using a first-order approximation, allowing us to reformulate the equations as follows:

$$\begin{aligned}\delta\dot{\boldsymbol{q}} &= \Delta t\left[\phi(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{\tau}) + \frac{\partial\phi}{\partial\boldsymbol{q}}\delta\boldsymbol{q} + \frac{\partial\phi}{\partial\dot{\boldsymbol{q}}}\delta\dot{\boldsymbol{q}}\right] \\ &= \Delta t\left[\phi(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{\tau}) + \frac{\partial\phi}{\partial\boldsymbol{q}}\Delta t(\dot{\boldsymbol{q}}(t) + \delta\dot{\boldsymbol{q}}) + \frac{\partial\phi}{\partial\dot{\boldsymbol{q}}}\delta\dot{\boldsymbol{q}}\right].\end{aligned} \tag{3.4}$$

Thus, with the joint velocities for the next time step computed, we can proceed to calculate the joint angles at the next time step using an explicit Euler time integration.

3.2.2 Muscle System

The Hill-type muscle model, first proposed by (Hill, 1938), is a fundamental component in biomechanical modeling. It provides a concise but effective representation of muscle mechanics by assuming that there are two primary sources of muscle force: the parallel element (PE), which passively generates a restoring force f_p due to the material elasticity of the muscle, and the contractile element (CE), which actively generates a contractile force f_c in response to motor neuron excitation. The total muscle force is $f_m = f_p + f_c$. More details of this popular model are presented by Ng-Thow-Hing (2001) and Winters (1990).

We use a modified version of the Hill-type muscle model for our biomechanical virtual human, which is a good balance of biomechanical accuracy and computational efficiency. We represent the passive element (PE) as a uniaxial exponential spring, given by the following equation:

$$f_p = \max\left(0, k_s\left(e^{k_c e} - 1\right) + k_d \dot{e}\right), \tag{3.5}$$

where k_s and k_c are the *elastic coefficients*, and k_d is the *damping coefficient*. The strain e is defined as $e = (l - l_0)/l_0$, where l denotes the current muscle length and l_0 denotes the slack muscle length. The *strain rate* $\dot{e}$ is given by $\dot{e} = \dot{l}/l_0$.

Obviously, only the state of the muscle system determines the passive element force f_p. Therefore, in Equation 3.1 we treat the passive element as part of $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ instead of the input part on the right.

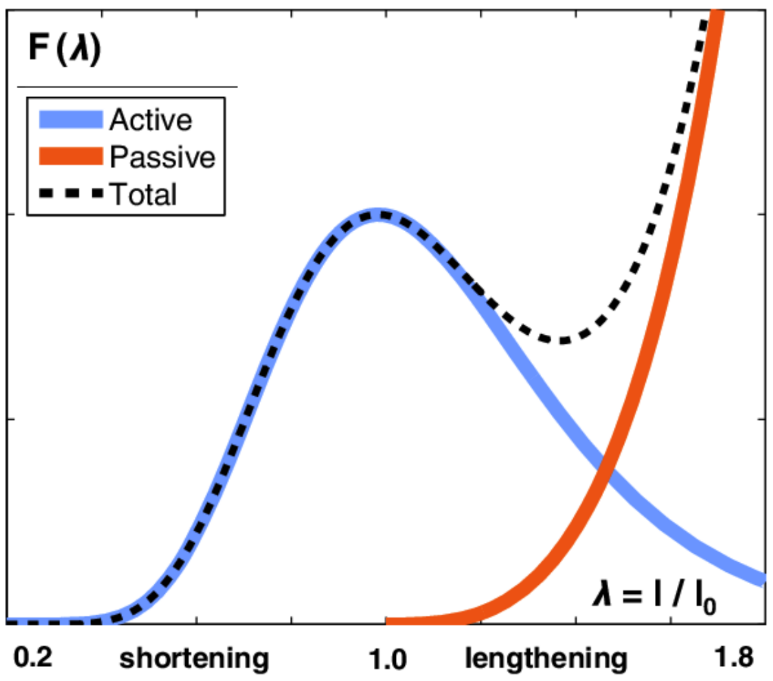


Figure 3.3: Relationship between total muscle force and stretch ratio in the Hill-type muscle model. The total muscle force reaches its maximum and then decreases following the active force (the contractile element), but then grows rapidly following the passive force (the parallel element). Plot from (Wisdom et al., 2015).

The contractile element (CE), on the other hand, is proportional to the activation level of the muscle and can be formulated as:

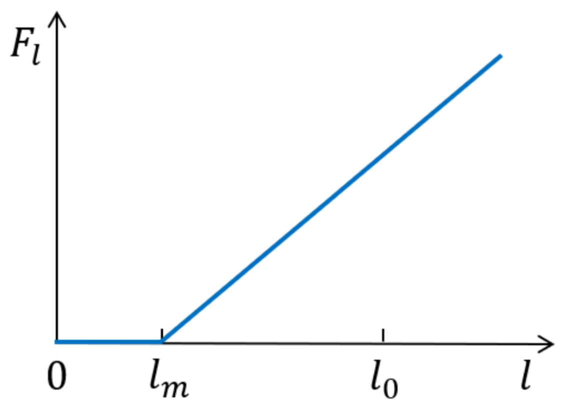
$$f_c = aF_l(l)F_v(\dot{l}), \tag{3.6}$$

$$F_l(l) = \max\left(0, k_{\max}(l - l_m)\right), \tag{3.7}$$

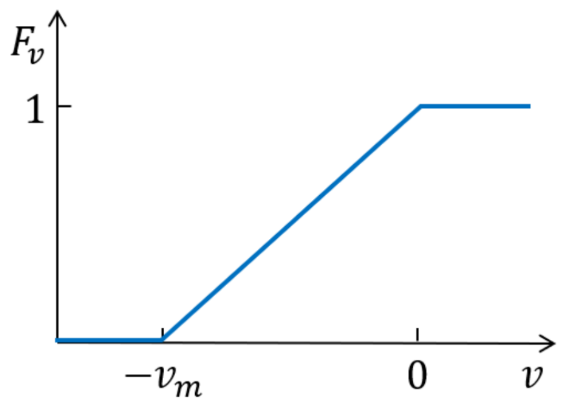
$$F_v(\dot{l}) = \max\left(0, 1 + \min(\dot{l}, 0)/v_m\right), \tag{3.8}$$

where $a \in [0, 1]$ is the muscle *activation level*, and F_l and F_v are the *force-length relation* and *force-velocity relation*, respectively.

The plots of the two relations are presented in Figure 3.4, and additional details about the parameter settings and the biomechanical background are given by Lee and Terzopoulos (2006). In brief, $k_{\max}$ is the maximum stiffness of a fully activated muscle, l_m is the minimum length at which the muscle can produce force, and v_m is the velocity threshold under which the contractile force can be exerted.



(a) force-length relationship

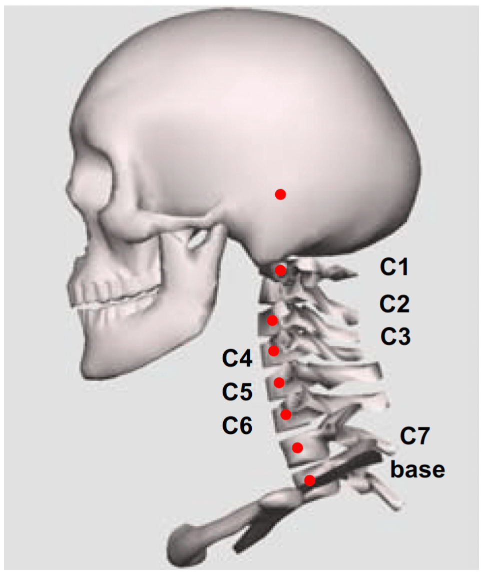


(b) force-velocity relationship

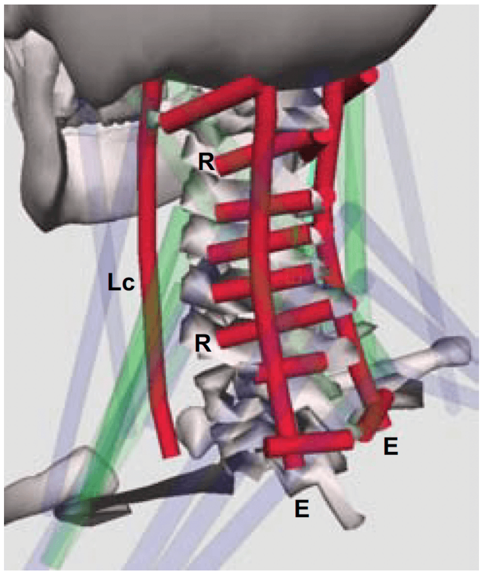
Figure 3.4: Plots illustrating the force relationships of the Hill-type muscle model. The force-length relationship (a) shows the proportional force increase with muscle length beyond the minimum muscle length l_m, while the force-velocity relationship (b) demonstrates the inverse relationship between force output and contraction velocity.

In most of our experimental configurations, we set the coefficient k_c to 7 for all muscles, and l_m is set to $l_0/2$ and v_m to $l_0\,\mathrm{s}^{-1}$. The coefficients related to muscle stiffness, such as k_s, k_d in the passive element (PE) and $k_{\max}$ in the contractile element (CE), are scaled to be proportional to the strength of each muscle, which is approximated by the physiological cross-sectional area (PCSA) of the muscle geometry. A detailed table regarding these weights is found in (Lee et al., 2009).

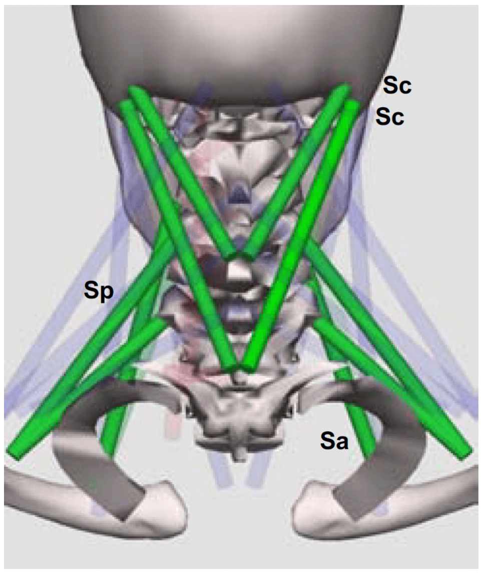
As shown in Figure 3.1, the full-body musculoskeletal structure consists of a total of 850 muscles with more than 2,500 force points. Depending on the specific use case, only a portion of the muscles in the area of interest will be enabled. In the bipedal motion experiments, we focus on the lower body and enable 78 muscles to capture the necessary details for high-fidelity simulations. On the other hand, our facial expression transfer experiments utilize the face-head-neck complex subset of the musculoskeletal model, as illustrated in Figure 3.5. This subset comprises 72 active cervical muscles that control the orientation of the head and face.



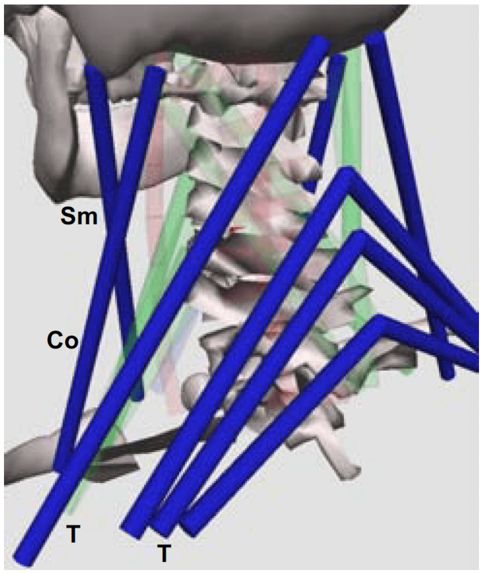
(a) Neck skeletons



(b) Deep muscles



(c) Intermediate muscles



(d) Superficial muscles

Figure 3.5: Detail of the musculoskeletal structure of the face-head-neck complex as specified in (Lee and Terzopoulos, 2006). (a) The red dots represent the pivots of the eight cervical vertebrae (C1-C7) and the skull. The neck muscle system consists of (b) 48 deep muscles, (c) 12 intermediate muscles, and (d) 12 superficial muscles.

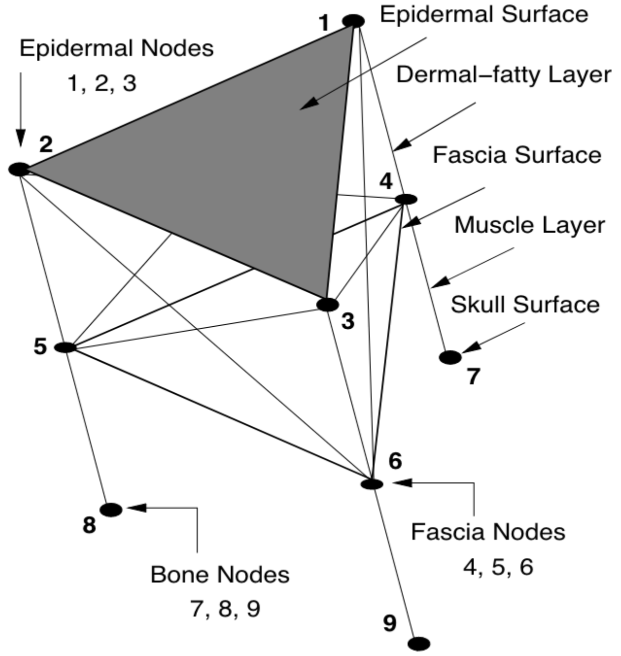


Figure 3.6: The triangular prism elements in the layered synthetic tissue model. The skin is made up of four layers—epidermis, dermis, sub-cutaneous connective tissue, and fascia—together with the fifth layer that includes the muscles of facial expression. Figure from (Lee et al., 1995).

3.3 Facial Soft Tissue

We adopt the physics-based facial tissue model introduced by Lee et al. (1995) for experiments requiring soft tissue simulation for delicate facial expression animation.

In this scheme, a structured generic face mesh is adapted to the laser-scanned facial images using a feature-based deformable model matching algorithm that fits the mesh to the labeled facial contours. This captures the facial geometry and provides texture coordinates for mapping the color image onto the mesh triangles. As a result, the reconstructed face mesh represents the relaxed, neutral expression as the base.

To dynamically simulate facial expressions, the original scheme constructs a layered synthetic tissue model consisting of the epidermis, dermis, sub-cutaneous connective tissue, and fascia layers. The tissue model is composed of triangular prism elements that match the adapted facial mesh triangles, with the nodes of each layer connected by spring structures, as illustrated in Figure 3.6. In addition, synthetic muscle actuators (linear and piecewise linear) are embedded in the muscle layer and contribute to skin deformation through their contractions.

The physical simulation of the muscle-actuated facial skin model is implemented as a discrete deformable model (DDM), where a network of fascia nodes is connected using uniaxial springs. The force exerted from spring j on node i is given by

$$\boldsymbol{g}_i^j = c_j(l_j - l_j^r)\boldsymbol{s}_j, \tag{3.9}$$

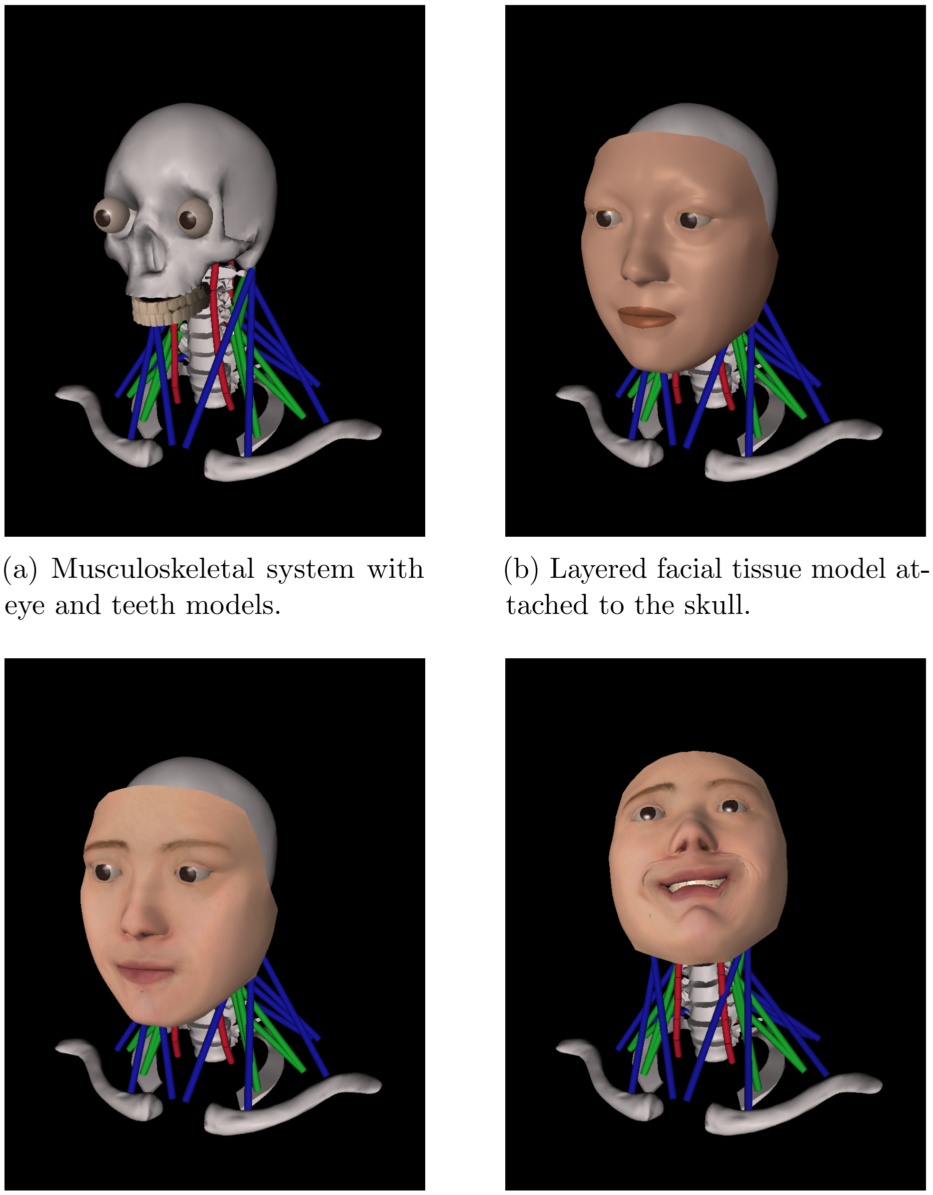
where l_j and l_j^r are the current and resting lengths of spring j, respectively, and $\boldsymbol{s}_j = (\boldsymbol{x}_j - \boldsymbol{x}_i)/l_j$ is the spring direction vector. The coefficient c_j is the *stress-strain relationship*, and each layer has its own value.

The underlying synthetic muscle fibers also actuate the facial skin model. The force $\boldsymbol{f}_i^j$ exerted from muscle j on node i is calculated according to the length scaling function Θ_1 and the muscle-width scaling function Θ_2 as follows:

$$\boldsymbol{f}_i^j = \Theta_1(\varepsilon_{j,i})\Theta_2(\omega_{j,i})\boldsymbol{m}_j, \tag{3.10}$$

where $\boldsymbol{m}_j$ is the normalized muscle vector for muscle j; Θ_1 scales the force according to the distance ratio $\varepsilon_{j,i} = \rho_{j,i}/d_j$, with d_j the muscle length; and Θ_2 scales the force according to the width ratio $\omega_{j,i}/w_j$, with w_j the muscle width.

Figure 3.7 demonstrates the simulated facial model integrated with the face-head-neck biomechanical complex. The structured facial soft-tissue mesh is attached directly to the skull's surface, with the underlying face-head-neck system driven by the Hill-type muscle actuators, which we have detailed in Section 3.2.2. To improve simulation efficiency, we decouple the muscle influences on soft tissue deformation from the articulation within the rigid body system.



(a) Musculoskeletal system with eye and teeth models.

(b) Layered facial tissue model attached to the skull.

(c) High-fidelity facial texture applied.

(d) Neuromuscular control for expressions and face orientation.

Figure 3.7: An overview of the neuromuscular simulation and control of the facial model.

3.4 Neuromuscular Control Frameworks

Exploring the capacity of neuromuscular control in biomechanical simulations requires robust and versatile frameworks. In our research, we leverage three such systems, each selected for its unique capabilities and advantages to specific experimental needs. This section delves into the characteristics of each framework—the “Heads Up” neuromuscular simulation and control framework for facial expressions, an OpenSim-based framework for prototyping reinforcement learning concepts in locomotion, and the modern Omniverse framework powered by the PhysX physics simulator for enhanced computational speed and scalability. We will discuss their implementation, advantages, and the limitations we encountered, providing insight into the evolving demands of computational biomechanics.

3.4.1 The “Heads Up” Face-Head-Neck Neuromuscular Framework

For facial expression simulation and transfer experiments, we adopted an improved version of the “Heads Up” face-head-neck neuromuscular simulation framework introduced in (Lee and Terzopoulos, 2006). This framework is implemented in native C++ and integrates the skeletal, muscular, and facial soft-tissue structures previously discussed. Leveraging Featherstone’s algorithm and backward Euler time integration, it facilitates semi-real-time animation of the face-head-neck complex on ordinary computing hardware, making it particularly suitable for our learning-based expression transfer experiments.

While effective for facial simulations, one of the limitations of this framework is its computational efficiency when extending to more complex body models. Several works, such as (Si et al., 2014), (Zhou, 2019), and (Nakada et al., 2018), have extended this framework to full-body biomechanical virtual human models. Figure 3.8 shows an example of such extension. However, these full-body simulations operate significantly slower than real-time, especially when inverse dynamics-based reflex control is enabled. The root cause of this efficiency issue lies in the framework’s inability to parallelize computations, therefore confining the simulation on the CPU in a single-thread manner.

As computational hardware techniques advance, physical simulation frameworks are

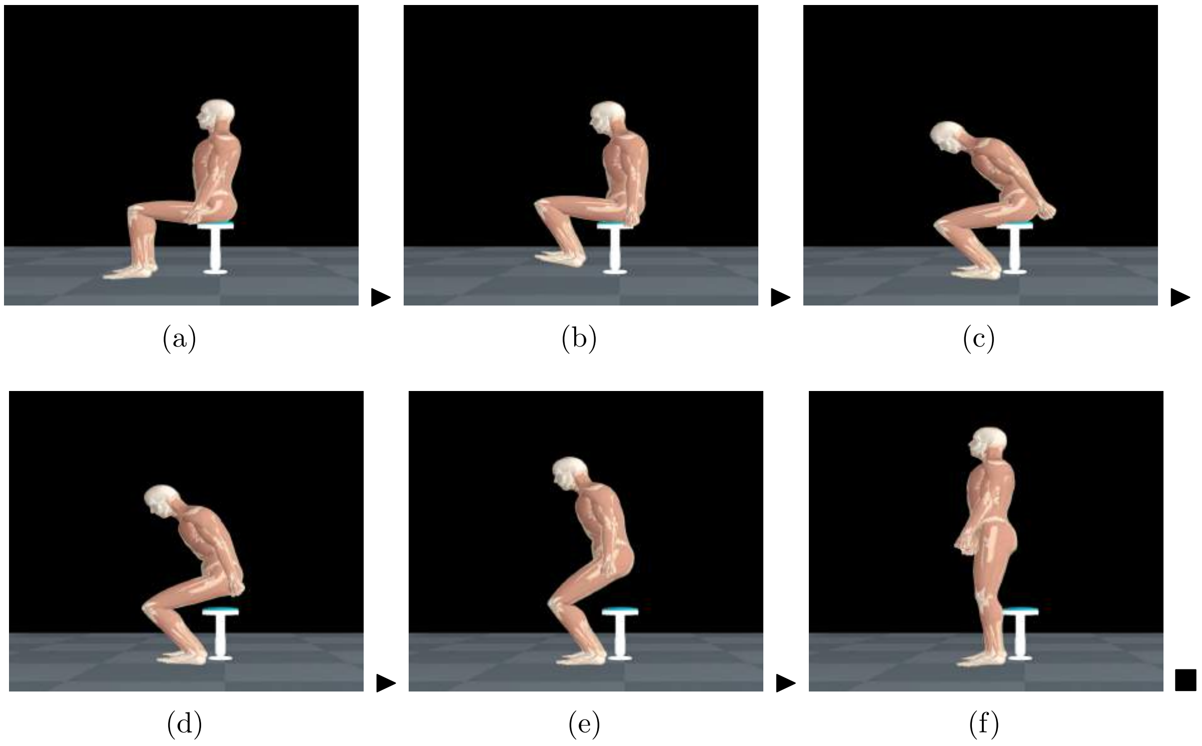
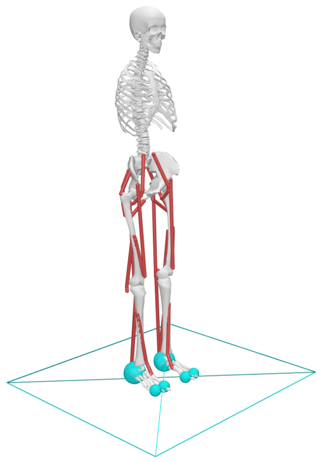


Figure 3.8: The full-body extension of the "Heads Up" face-head-neck neuromuscular framework introduced in (Zhou, 2019). In this example, the human model stands up from a seated position on the chair and remains standing through muscle activation control.

increasingly optimized for parallel computing devices such as GPUs.

3.4.2 OpenSim-Based Neuromuscular Framework

To investigate the potential of reinforcement learning for tuning the CPG controller for human locomotion, with a particular focus on bipedal locomotion, we used a modified implementation from the "Learn to Move" challenge of the NeurIPS 2019 competition track (Kidziński et al., 2018), which is based on OpenSim (Seth et al., 2018). OpenSim provides a powerful platform for studying neuromuscular systems, and the "Learn to Move" framework builds on this foundation by incorporating a convenient interface for reinforcement learning. This allows us to leverage the biologically realistic modeling provided by OpenSim while seamlessly integrating it with state-of-the-art reinforcement learning techniques. The framework uses a simplified full-body skeleton without arms and removes the upper body degrees of freedom, as shown in Figure 3.9, making it lightweight


(a) Simplified model

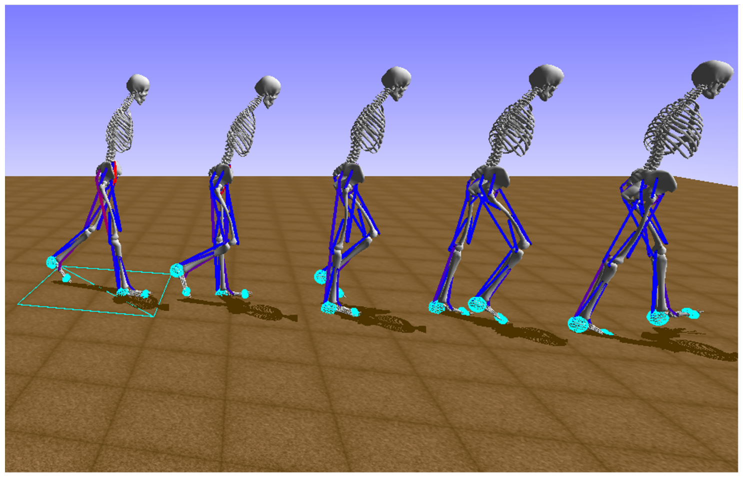

(b) Simulation in OpenSim

Figure 3.9: The OpenSim-based simulation framework with a simplified full-body model. The skeleton and muscles are shown in (a). Tthe arms are omitted to reduce complexity and computational load. (b) A simulation of the model stepping forward, with muscle activation levels indicated by color (red represents high activation; blue represents low activation).

in terms of memory consumption and computational load. As a result, we can speed up the data acquisition for the reinforcement learning procedure via multi-processing.

As for its limitations, compared to our model geometry, the Learn to Move framework has fewer lower body muscles and uses an oversimplified rigid body collision model for both internal interactions within the human body and external interactions with the ground. In addition, while the sampling efficiency is sufficient for multiprocessing reinforcement learning on CPUs, the framework does not currently take advantage of the computational power of GPUs. This leaves a bottleneck in terms of CPU performance and kernel count.

3.4.3 Omniverse With the PhysX Simulation Framework

To overcome the limitations of previous frameworks and take advantage of the latest advances in GPU-accelerated physics simulation, we explored NVIDIA's Omniverse platform with the PhysX simulation engine. Omniverse provides a unified environment for creating, simulating, and visualizing complex virtual worlds through an extension-based ecosystem. In particular, we used *Isaac Sim*, a powerful tool designed for physically

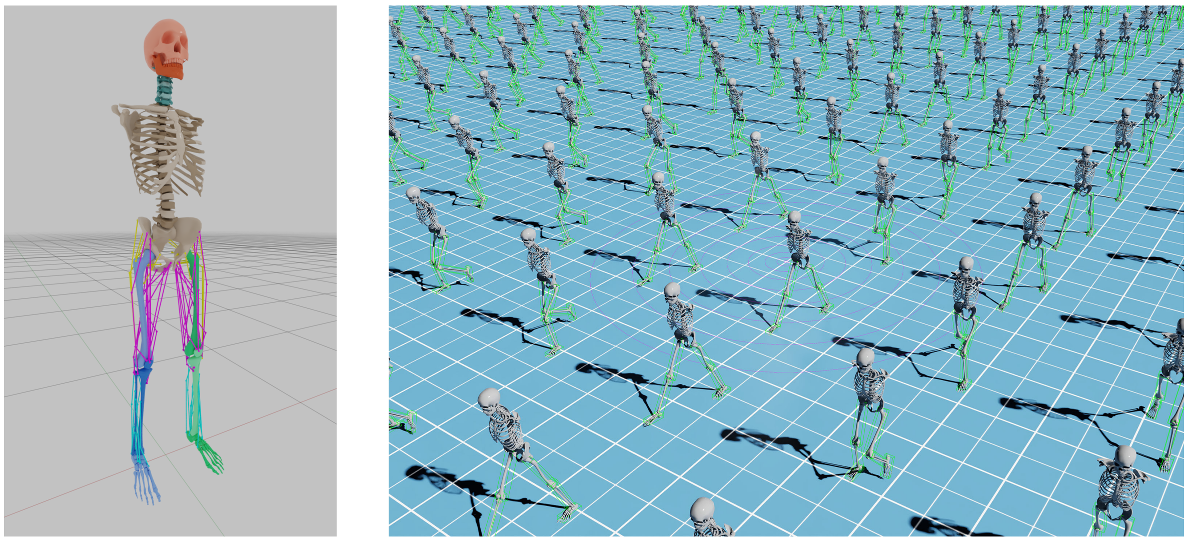

(a) Simplified model (b) Simulation in Omniverse Issac Sim

Figure 3.10: (a) The simplified human musculoskeletal model used in our reinforcement learning-based CPG control task, with arms removed and reduced degrees of freedom in the upper body. (b) A total of 512 instances of the model are simulated simultaneously in NVIDIA Omniverse Isaac Sim, enabling efficient trial and error for reinforcement learning.

accurate robotics simulation and synthetic data generation (NVIDIA Corporation, 2023). However, we discovered that Isaac Sim's capabilities extend beyond its primary domain, making it well-suited for our neuromuscular simulations for virtual human models.

We migrated our high-fidelity human model into the Isaac Sim environment and simplified it for our reinforcement learning-based CPG control task. As illustrated in Figure 3.10, to improve computational efficiency similar to the approach described above, we removed the arms and reduced the degrees of freedom in the upper body. Isaac Sim's native support for the Gym interface allows for seamless integration with reinforcement learning algorithms. Moreover, Isaac Sim's scalability, powered by the GPU-accelerated PhysX engine, enables the simultaneous simulation of hundreds or even thousands of instances of the model. Each instance serves as an independent learning agent, facilitating efficient exploration and learning. We leverage the computational capability of an NVIDIA RTX 4070 GPU with 12GB of RAM to achieve this level of parallelism.

CHAPTER 4

CPG-Based Bipedal Locomotion Control

This chapter investigates the synergistic combination of Central Pattern Generators (CPGs) and reinforcement learning techniques to achieve muscle-level locomotion control in a virtual human model. We formulate our problem as a model-free reinforcement learning task and detail our methodology, which includes a dynamical system-based representation of CPGs, multiple state-of-the-art deep reinforcement learning algorithms, and various experimental setups optimized for hardware-accelerated performance. Subsequently, we conduct extensive experiments and present a detailed analysis of our results.

4.1 Overview

Locomotion control for neuromuscular human models presents significant challenges, particularly when employing muscle activation-based control. Traditional optimization methods, such as inverse dynamics, can estimate the desired torque and muscle activation over time with the help of reference motion or retroreflective (infrared) markers (Sylvester et al., 2021). However, the human body has biological redundancy, resulting in an indeterminate system of physics-based equations due to the presence of more muscles than necessary to actuate joint movements. This makes it challenging to determine the optimal muscle activation patterns. Additionally, the muscle activation results computed from optimization algorithms are not necessarily biomimetic, may not guarantee natural motions, and struggle to adapt to changes in the environment.

Recent advances in deep learning have inspired researchers to address these challenges through imitation learning, where a controller is trained to mimic captured reference

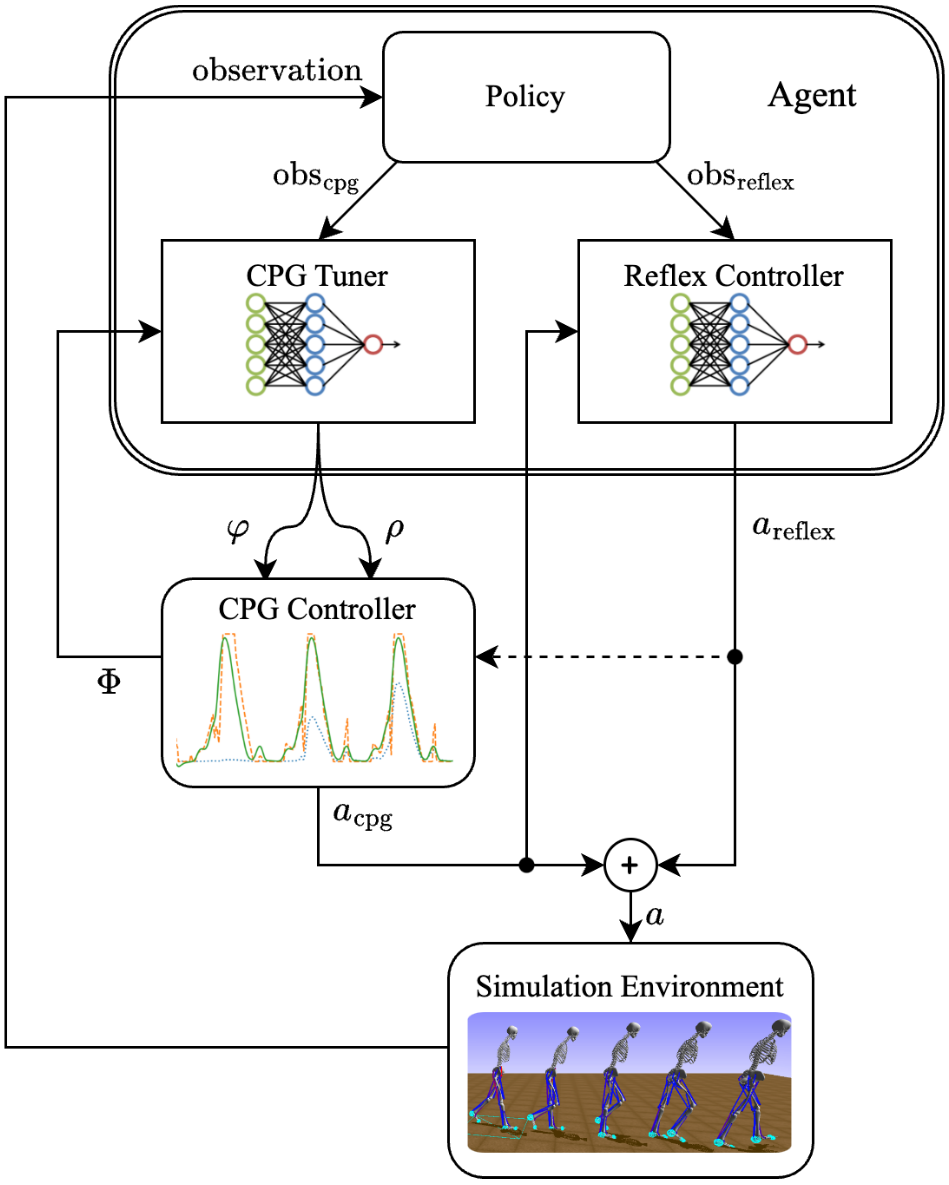


Figure 4.1: Overview of our proposed control framework for integrating deep reinforcement learning with a Central Pattern Generator (CPG) to enable adaptive and flexible locomotion control for a simulated environment. The learning agent receives observations and dispatches them to the CPG Tuner and Reflex Controller networks. The CPG Tuner outputs modulation signals φ and ρ to fine-tune the phase and amplitude of the CPG Controller, which generates rhythmic patterns of muscle activations a_{cpg}. These patterns are combined with reflex control signals a_{reflex} to produce the final muscle activation signals for the simulated virtual human model.

data or expert behavior. However, the controller may exhibit poor generalization to new environments and tasks that deviate from the training data. While the generated motions appear human-like, they are not necessarily optimized for the dynamics of the musculoskeletal model, as mentioned in (Song et al., 2021).

Deep reinforcement learning, on the other hand, has shown potential for dynamic and generalizable locomotion control. It allows controllers to refine their actions through direct interaction with the physical simulation environment, optimizing their behavior for the given task and dynamics in a more robust and adaptive manner. Due to the high-dimensional control space and local optima, controllers trained from scratch with deep reinforcement learning may converge to unnatural, inefficient gaits that are not human-like, even if they accomplish the task and maximize the reward. In addition, the computational requirements of deep reinforcement learning are substantial.

We propose a sophisticated methodology for constructing and training a locomotion controller that synergistically combines deep reinforcement learning with the traditional biomimetic approach of the Central Pattern Generator (CPG). Our focus is on bipedal locomotion control, such as walking and running. In our methodology, we first utilize expert behaviors to initialize the CPG parameters, allowing the human model to generate cyclic and naturally coordinated stepping movements via muscle activations. Deep reinforcement learning is then used to fine-tune the behavior of the CPG. Our approach enables the controller to produce stable and natural gaits that are capable of efficiently performing various tasks in dynamic simulation environments. Figure 4.1 presents an overview of our method.

In the following sections, we explain the methodologies underlying each component of our framework. This discussion is intended to provide a comprehensive understanding of the fundamental elements of our approach, emphasizing their collective contribution towards realizing adaptive and biomimetic locomotion control in virtual human models.

4.2 Central Pattern Generator

Central Pattern Generators (CPGs) are neuronal circuits typically located in the vertebrate spinal cord or brain stem area. These circuits play a crucial role in controlling various rhythmic motor patterns in mammals, such as breathing, swimming, and walking. The ability of CPGs to generate rhythmic signals without requiring rhythmic input has made them an attractive concept for robotics researchers seeking to replicate natural locomotion patterns.

4.2.1 Dynamical System Formulation

The CPG controller in our framework is implemented based on a nonlinear dynamical system introduced by (Gams et al., 2009). The system is designed to learn and encode periodic signals without prior knowledge of their frequency and waveform. Additionally, it modulates the learned periodic patterns in response to external events. The system consists of two main components: the Canonical Dynamical System (CDS), responsible for extracting the fundamental frequency of the input signal using adaptive frequency oscillators, and the Output Dynamical System (ODS), responsible for learning the waveform using a locally weighted regression (LWR) algorithm.

For clarity, we demonstrate the nonlinear dynamical system in a single degree of freedom (DOF). The trajectory y, which represents the output signal of our CPG, oscillates around the anchor point g based on the pattern determined by the following set of differential equations:

$$\dot{z} = \Omega \left(\alpha_z(\beta_z(g - y) - z) + \frac{\sum_{i=1}^{N} \Psi_i w_i r}{\sum_{i=1}^{N} \Psi_i} \right), \tag{4.1}$$

$$\dot{y} = \Omega z, \tag{4.2}$$

$$\Psi_i = \exp(h(\cos(\Phi - c_i) - 1)). \tag{4.3}$$

Here, $z(t)$ is an intermediate variable that describes the first derivative of the trajectory

y. The variable Φ denotes the phase of the signal, which is based on the fundamental frequency Ω provided by the Canonical Dynamical System (CDS). For periodic motions like walking, we can specify Ω as a constant, such as $\frac{2\pi}{T}$, where T is the period of the motion cycle determined by the reference motion or specified by the task. The positive constants α_z and β_z (both (Gams et al., 2009) and (Si et al., 2014) set them to $\alpha_z = 8$ and $\beta_z = 2$) ensure critical damping and monotonic variation of the system around the anchor point g. The number of Gaussian-like periodic kernel functions Ψ_i is denoted by N, and h determines the width of these kernel functions. We set $N = 25$ and $h = 2.5N$ for all our simulations, and c_i are equally spaced between 0 and 2π in N steps. The amplitude control parameter is r, which we initially set to 1.0.

4.2.2 Reference-Based CPG Pattern Learning

The trajectory pattern of the CPG within a full cycle is determined by the kernel weights w_i, and to initialize the weights based on reference data (denoted as y_{demo}), we can employ Incremental Locally Weighted Regression (ILWR) (Vijayakumar and Schaal, 2000), which minimizes the quadratic error criterion

$$J_i = \sum_{t=1}^{P} \Psi_i(t) \left(f_{targ}(t) - w_i r(t)\right)^2, \tag{4.4}$$

where t denotes the discrete time step, P denotes the number of time steps, which could be finite or infinite, and the target data points are formulated as

$$f_{targ} = \frac{1}{\Omega^2}\ddot{y}_{demo} - \alpha_z \left(\beta_z(g - y_{demo}) - \frac{1}{\Omega}\dot{y}_{demo}\right). \tag{4.5}$$

According to (Gams et al., 2009), the regression could either be performed as a batch regression or as incremental minimization of the error criterion J_i with a stream of reference signal over time. Since the reference motion we use may present perturbations and noise in different cycles, we prefer the latter as do the original authors. Specifically, given the target data $f_{\text{targ}}(t)$ and $r(t)$, the kernel weights w_i are updated iteratively by

$$w_i(t+1) = w_i(t) + \Psi_i P_i(t+1) r(t) e_r(t), \tag{4.6}$$

$$P_i(t+1) = \frac{1}{\lambda}\left(P_i(t) - \frac{P_i(t)^2 r(t)^2}{\frac{\lambda}{\Psi_i} + P_i(t) r(t)^2}\right), \tag{4.7}$$

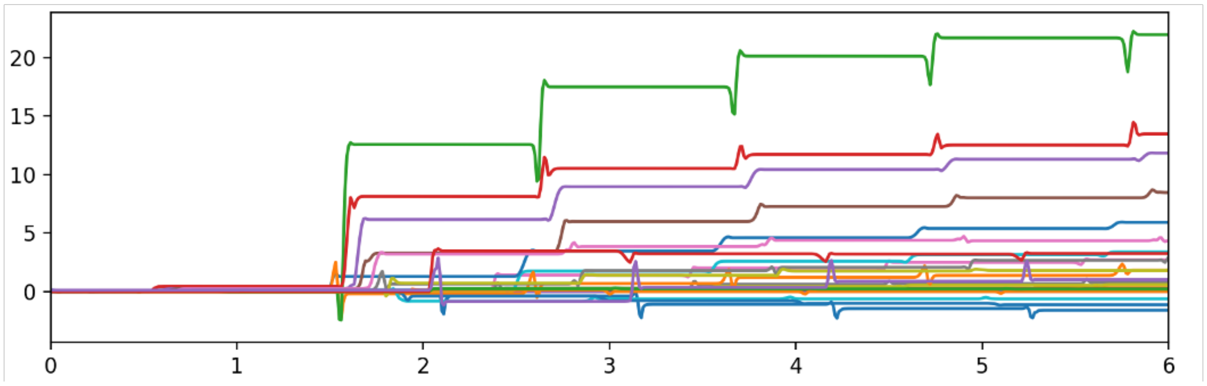
$$e_r(t) = f_{\text{targ}}(t) - w_i(t) r(t). \tag{4.8}$$

In these equations, P_i is the inverse covariance matrix introduced by (Ljung and Söderström, 1983), λ is the forgetting factor, and e_r is the weight learning error. The incremental optimization process starts with $w_i = 0$ and $P_i = 1$. When the forgetting factor λ is set to 1, batch learning regression and incremental learning regression will obtain the same weights w_i for the same training set. However, when the forgetting factor is less than 1, incremental regression will give more weight to the most recent data, effectively tending to forget older data. We set this factor ranging from 0.95 to 0.99 based on different sources of reference motions. Figure 4.2 demonstrates an example of the learning procedure with the above method.

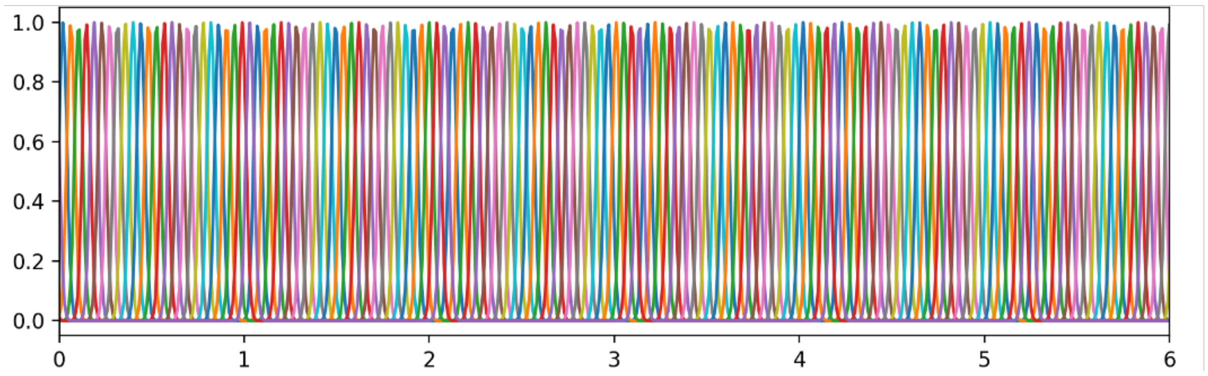
4.2.3 Modulation of the CPG Signals

As shown in Figure 4.1, our trained CPG acts as a separate controller that outputs muscle activation signals (a_{cpg}) that directly actuate the muscles in our simulated environment. The policy does not determine the phase or pattern of the CPG directly but uses intermediate modulation signals (φ and ρ) to tune the CPG controller.

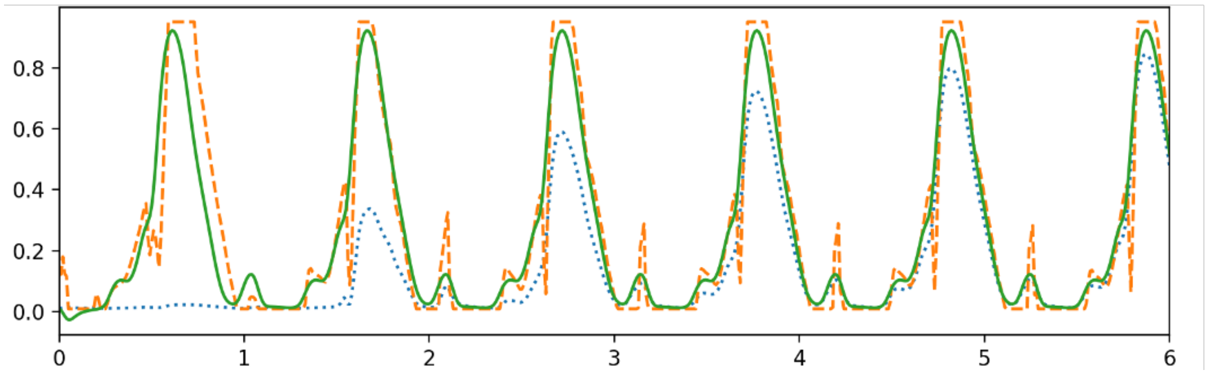
The *frequency multiplier* φ (ranging from 0.25 to 1.75 in most of our experiments) will modulate the phase velocity of the CPG, and the *amplitude multiplier vector* ρ with each ρ_i ranging from 0 to 1.0 will modulate the strength of each muscle activation signal. In other words, φ is applied to Ω and ρ_i is applied to w_i. The modulated version of



(a) Changing of kernel weights



(b) Value of kernel functions over time



(c) Learned signal and the reference

Figure 4.2: An example of the learning procedure of the trajectory pattern using Incremental Locally Weighted Regression (ILWR). The modification of the kernel weights and kernel function values over time is demonstrated in (a) and (b). In (c), the dashed orange line is the reference signal to learn, the dotted blue line is the output signal during the learning process, showing the current learned pattern at each time step, and the solid green line represents the final learned signal after the learning procedure converges.

Equation 4.1 may be expressed as

$$\dot{z} = \Omega_{\text{tuned}} \left(\alpha_z(\beta_z(g-y)-z) + \frac{\sum_{i=1}^{N} \Psi_i w_i \rho_i r}{\sum_{i=1}^{N} \Psi_i} \right), \tag{4.9}$$

where the modulated fundamental frequency is

$$\Omega_{\text{tuned}} = (1+\varphi) \times \Omega_{\text{original}}. \tag{4.10}$$

4.3 Deep Reinforcement Learning

Reinforcement learning (RL) is a framework for learning and decision-making in which an agent interacts with an environment and learns to maximize a cumulative reward signal through trial and error. Our proposed locomotion control scheme employs deep reinforcement learning algorithms to optimize a policy that modulates and adapts the Central Pattern Generator (CPG) controller to maintain stable motions in complex scenarios.

As shown in Figure 4.1, our reinforcement learning agent optimizes a policy consisting of two neural networks: the *CPG Tuner* and the *Reflex Controller*. The *CPG Tuner* outputs modulation signals to adjust the CPG controller, while the *Reflex Controller* generates muscle activation adjustment signals in response to the environment. These networks are parameterized by deep neural networks, allowing the agent to learn sophisticated locomotion behaviors from high-dimensional observations and adapt to diverse tasks and dynamic environments.

The reinforcement learning framework may be formalized as a Markov decision process (MDP) consisting of the following components (Arulkumaran et al., 2017):

- A series of states $\boldsymbol{s}_t \in \mathcal{S}$ representing the status or configuration of the environment over time. At the beginning, the RL agent samples an initial state according to the distribution of the start state $\boldsymbol{s}_0 \sim p(\boldsymbol{s}_0)$.

- A series of actions $\boldsymbol{a}_t \in \mathcal{A}$ denoting how the agent reacts to the environment, which are sampled from its policy $\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$.
- Transition dynamics $\mathcal{T}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$ that describe how the environment evolves from one state to another based on the agent's actions, *i.e.*, $\boldsymbol{s}_{t+1} \sim \mathcal{T}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$.
- A reward function $\mathcal{R}(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$ that determines the rewards r_t received by the agent, serving as the optimization target of the policy.
- A discount factor $\gamma \in [0, 1]$ that determines the priority of short-term and long-term rewards. A lower discount factor places more emphasis on immediate rewards.

The objective of the RL agent is to discover an optimal policy π^* that maximizes the expected cumulative reward $J(\pi)$ over time. The expected cumulative reward, also known as the objective function of the policy, is defined as the expectation of the sum of discounted rewards over a finite or infinite horizon:

$$\pi^* = \arg\max_{\pi} J(\pi). \tag{4.11}$$

Here,

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right], \tag{4.12}$$

where $\tau = (\boldsymbol{s}_0, \boldsymbol{a}_0, r_0, \boldsymbol{s}_1, \boldsymbol{a}_1, r_1, \dots)$ represents a *trajectory* of states, actions, and rewards sampled from the policy π, whose distribution $p(\tau|\pi)$ may be induced as

$$p(\tau|\pi) = p(\boldsymbol{s}_0) \prod_{t=0}^{T} \mathcal{T}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)\pi(\boldsymbol{a}_t|\boldsymbol{s}_t). \tag{4.13}$$

4.3.1 Actor-Critic Methods

Although basic optimization methods such as the Policy Gradient can be effective in straightforward environments by directly optimizing the policy based on the gradient of the expected reward with respect to the policy parameters, they can be problematic in more complex settings due to high variance in the gradient estimates, resulting in unstable learning.

The use of actor-critic methods has proven advantageous in complex reinforcement learning environments. These techniques divide the learning process into two specific parts: the actor, in charge of adjusting the policy parameters through exploration and exploitation, and the critic that predicts the future rewards for specific states or state-action pairs using value functions. The critic's estimates serve as a baseline for the actor's policy updates, helping to reduce the variance in the gradient estimates. By decoupling policy parameters and value function optimization, actor-critic methods can stabilize the learning process and improve the overall performance of the reinforcement learning agent.

There are two main types of value functions for the critics:

- **State Value Function**, typically denoted as $V^{\pi}(\boldsymbol{s})$, which estimates the expected cumulative reward starting from a given state $\boldsymbol{s}$ and then following the policy π:

$$V^{\pi}(\boldsymbol{s}) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T-1}\gamma^t r_t | s_0 = \boldsymbol{s}\right]. \tag{4.14}$$

- **State-Action Value Function**, typically denoted as $Q^{\pi}(\boldsymbol{s}, \boldsymbol{a})$ and also known as the *Q-function*, which estimates the expected cumulative reward starting from a given state $\boldsymbol{s}$, taking action $\boldsymbol{a}$, and then following the policy π:

$$Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T-1}\gamma^t r_t | s_0 = \boldsymbol{s}, a_0 = \boldsymbol{a}\right]. \tag{4.15}$$

The actor-critic framework has been successfully applied in various state-of-the-art reinforcement learning algorithms, such as Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC). These algorithms employ the actor-critic architecture in different ways to address specific challenges in reinforcement learning, such as ensuring stability and robustness in policy updates, or encouraging exploration and reducing bias in value estimates.

The subsequent sections will examine how PPO and SAC implement the actor-critic framework and their distinct characteristics and benefits in the context of our research on virtual character locomotion control.

4.3.2 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a state-of-the-art actor-critic algorithm that has gained popularity due to its simplicity, stability, and strong performance across a wide range of reinforcement learning tasks (Schulman et al., 2017). PPO was developed to address the shortcomings of previous actor-critic methods, such as Trust Region Policy Optimization (TRPO) (Schulman et al., 2015a), which can be computationally expensive and difficult to implement.

The clipped surrogate objective is at the core of PPO. Its purpose is to stabilize policy updates by limiting the size of the policy change at each step. The objective is defined as

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right], \tag{4.16}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio between the new and old policies, and ϵ is a hyperparameter that controls the clipping range (usually set to 0.2).

The actor in PPO is represented by a policy network that outputs the mean and standard deviation of a Gaussian distribution over actions, from which the actual actions are sampled. The critic is a value function network $V^\pi(\boldsymbol{s})$ that estimates the expected cumulative reward from a given state. The advantage estimates $\hat{A}_t$ are computed using Generalized Advantage Estimation (GAE) (Schulman et al., 2015b), which balances the trade-off between bias and variance in the advantage estimates.

During training, PPO switches between collecting experience using the current policy and updating the policy using the clipped surrogate objective. The hyperparameters and network architectures can be tuned to optimize performance for specific tasks. In our research, PPO is one of the primary algorithms for locomotion controller learning due to its stability and robustness. We use PPO in both OpenSim-based and Omniverse-based simulation environments.

However, one drawback of PPO is its sample efficiency compared to other algorithms, such as Soft Actor-Critic (SAC). PPO requires a larger number of samples to achieve optimal performance, which can be a limitation in computationally expensive environments. This issue is particularly relevant in the OpenSim-based simulation environment, which suffers from constrained parallelism and limited hardware acceleration for efficient sample acquisition. In contrast, the Omniverse-based environment, powered by the GPU-accelerated PhysX engine, allows for more efficient sample collection through the simultaneous simulation of hundreds of instances of our human neuromuscular model.

4.3.3 Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) is an off-policy actor-critic algorithm that aims to maximize both the expected cumulative reward and the entropy of the policy (Haarnoja et al., 2018). By incorporating entropy maximization into the objective function, SAC encourages exploration and learns versatile policies that can adapt to different tasks and dynamic environments, making it suitable for continuous control tasks like our environments.

The SAC algorithm is based on the maximum entropy reinforcement learning framework. This framework augments the standard RL objective (as in Equation 4.12) with an entropy term

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T}\gamma^t r_t + \alpha\mathcal{H}(\pi(\cdot|\boldsymbol{s}_t))\right], \tag{4.17}$$

where $\mathcal{H}(\pi(\cdot|\boldsymbol{s}_t))$ is the entropy of the policy at state $\boldsymbol{s}_t$, which encourages the selection of diverse actions. The temperature parameter α, which controls the trade-off between maximizing reward and maximizing entropy, can be adaptively adjusted to maintain a desired level of entropy.

Similar to PPO, the actor in SAC is a policy network that outputs the mean and standard deviation of a Gaussian distribution over actions. However, the critic in SAC consists of two Q-function networks that estimate the expected cumulative reward plus the entropy term. The use of two independently updated Q-functions helps mitigate positive bias in the value estimates by taking the minimum of the two estimates. Additionally, SAC employs a separate value function network that computes entropy-adjusted value estimates for policy training.

SAC employs a soft policy iteration scheme that alternates updates between the Q-functions and the policy. The Q-functions are updated using a combination of Bellman backups and entropy regularization (Sutton and Barto, 2018), which helps ensure they remain consistent with the current policy while integrating the entropy term. This consistency is crucial as it encourages exploration and aims to maximize the expected cumulative reward. The policy itself is updated based on these Q-function estimates.

In our research, we employ SAC as an alternative algorithm for the OpenSim-based simulation framework. The sample efficiency and robustness of SAC, combined with its ability to efficiently leverage past experience and adaptively balance exploration with reward maximization, make it particularly well-suited for this environment, where computational costs are high and sample budgets are limited.

4.3.4 Architecture

The architecture of the reinforcement learning agent plays a significant role in its performance, efficiency, and the resources needed to learn the optimal policy. Additionally, a well-designed architecture can improve the agent's ability to adapt to new environments. In this section, we present the architecture of our proposed deep reinforcement learning agent for locomotion control, which employs a dual policy network approach to separate the Central Pattern Generator (CPG) modulation signals and muscle activation adjustments from the reflex control.

Our proposed architecture, as illustrated in Figure 4.3, consists of two main components: the *CPG Tuner* and the *Reflex Controller*, both of which are fully connected neural networks. The pipeline takes the current state of the virtual human $\boldsymbol{s}$, which includes

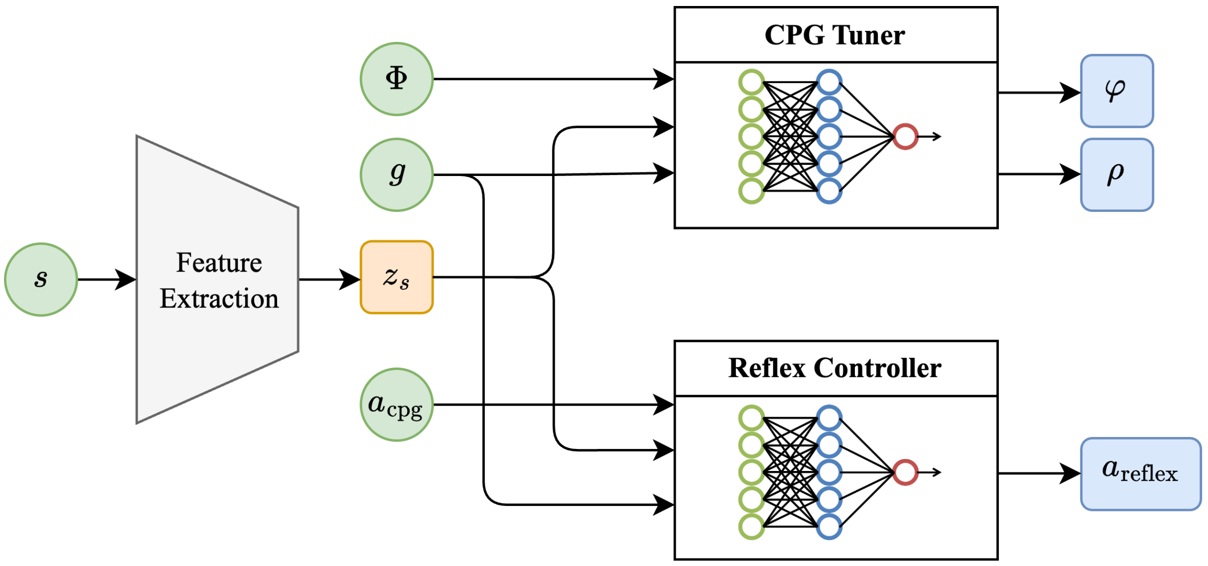


Figure 4.3: The proposed architecture for locomotion control, featuring a dual-module design with a CPG Tuner and a Reflex Controller. The shared feature extraction module efficiently processes the body state $\boldsymbol{s}$ into a latent space representation $\boldsymbol{z_s}$. Φ is the current phase of the CPG, $\boldsymbol{g}$ is the goal representation, and $\boldsymbol{a}_{\text{cpg}}$ is the current CPG output signals, which are muscle activations.

information such as joint angles, velocities, and muscle states, and passes it through a shared *Feature Extraction* module. This module transforms the state into a latent space representation, resulting in a state feature vector $\boldsymbol{z_s}$. In the next stage, the feature vector is concatenated with different auxiliary information and fed as input to the two networks. Each network processes the input and generates specific control signals based on its designated role in the control scheme.

The CPG Tuner is responsible for modulating the CPG signals based on the current body state, represented by the feature vector $\boldsymbol{z_s}$, the CPG phase Φ, and the goal representation $\boldsymbol{g}$. By adjusting the frequency and amplitude of the CPG controller through the φ and ρ signals, respectively, the CPG Tuner adapts the generated motion patterns to the desired task and environmental conditions. The self-contained nature of the CPG signals encourages the generation of natural motions that are consistent with biological principles.

The Reflex Controller, on the other hand, generates muscle activation adjustments $\boldsymbol{a}_{\text{reflex}}$ based on the current body state in the form of feature vector $\boldsymbol{z_s}$, CPG controller output $\boldsymbol{a}_{\text{cpg}}$, and goal representation $\boldsymbol{g}$. These adjustments serve as reflexive responses to perturbations and deviations from the desired motion, enabling the virtual human to maintain balance and stability. The Reflex Controller complements the CPG Tuner by providing real-time feedback-based adaptations to the muscle activation signals. When manually enabled, these adjustments can be used to fine-tune the CPG kernel weights for better output patterns, allowing for further refinement of the generated motions.

This architectural design incorporates domain expert knowledge of human biomimetics and hierarchical locomotion. The shared feature extraction module allows the CPG Tuner and Reflex Controller networks to utilize a common latent space representation of the body states, which reduces computational redundancy and encourages consistent learning across the two networks. Meanwhile, the synergistic combination of the rhythmic CPG signals and the feedback-based reflex signals creates a control system that exploits the inherent stability and naturalness of the CPG while allowing flexible adjustment to new circumstances.

Our experiments demonstrate that this dual-modular design is advantageous for learning more natural motions and accelerating the initial stages of the training process, which allows the agent to efficiently improve its performance and achieve longer episode lengths compared to alternative architectures.

4.4 Experimental Setup

This section outlines the two primary configurations of simulation environments utilized in our experiments. As previously discussed in Section 3.4, for the reinforcement learning experiment on locomotion control, we employed the OpenSim-based environment and the Omniverse-based environment. Since we have already discussed their characteristics and comparisons regarding their physics simulation backends and simulation efficiency, in this section we will focus primarily on the details of the virtual human models utilized in the two setups. We will discuss the degrees of freedom, observation and action spaces, as well as the amount and distribution of muscles associated with each model. Additionally, we

Component	Variables	Dimensions
Pelvis state	Pelvis position	3
	Pelvis linear velocity	3
	Pelvis angular velocity	3
Joint states (per side)	Joint positions (hip×2, knee, ankle)	4×2
	Joint velocities (hip×2, knee, ankle)	4×2
Ground forces (per side)	Force sensors on the feet (toe×2, heel)	3×2
Muscle states (per side)	Muscle force	11×2
	Muscle length	11×2
	Muscle velocity (change rate of length)	11×2
Total		97

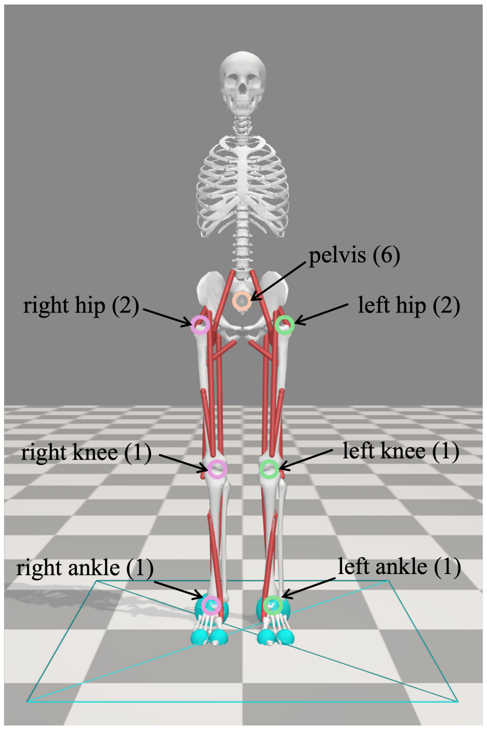
Table 4.1: Observation space components in the OpenSim-based environment

detail the configuration of the reward design, initial state distribution, as well as the early termination policy and how they contribute to successful learning.

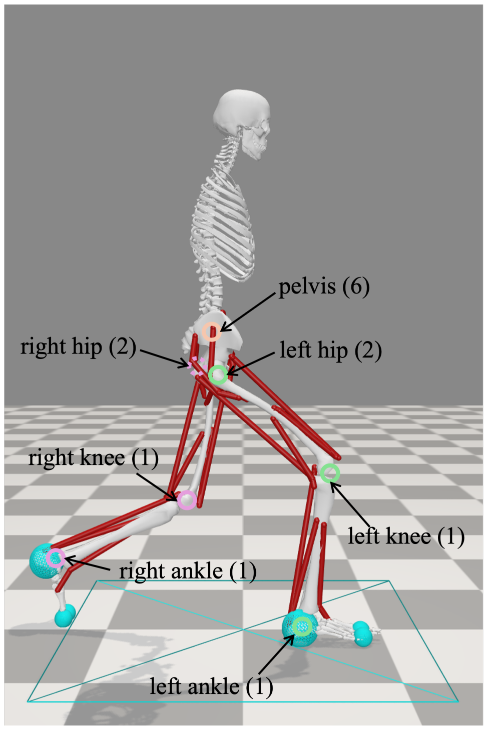
4.4.1 Observations and Actions

The observation spaces in the two major simulation environments differ due to the variations in the musculoskeletal models employed. Table 4.1 provides a detailed breakdown of the body state observation space dimensions for the simplified musculoskeletal human model in our OpenSim-based simulation environment. The observation space consists of a total of 97 dimensions, consisting of crucial information about the pelvis, joints, ground reaction forces, and muscle states.

The design of this observation space is a modified version based on the configuration in (Kidziński et al., 2018). The pelvis state is represented by 9 variables, including position, linear velocity, and angular velocity, which capture the global movement of the model. Each side of the legs contains 4 DOFs for the joint angles, determining the local lower body configuration, and 11 muscles, each with length, velocity (rate of length change), and force states, providing detailed information about the individual muscle behavior. Additionally, 3 ground reaction force sensors on each foot provide essential data about



(a) Frontal view



(b) Left-side view

Figure 4.4: Degrees of freedom of the pelvis and joints in the observation space of the OpenSim-based model.

the interaction between the model and the environment. Figure 4.4 illustrates these dimensions and joint DOFs.

The action space in our CPG-based locomotion control framework differs from traditional action spaces in reinforcement learning-based control schemes, where each muscle exactly matches an action signal. As previously shown in Figure 4.3, our action space is composed of the outputs from the CPG Tuner and the Reflex Controller. The CPG Tuner generates two types of signals: a scalar frequency multiplier φ and a vector ρ of amplitude multipliers, with each element corresponding to an individual muscle. And the Reflex Controller outputs muscle activation adjustment signals for each muscle. The total dimension of the action space for a specific simulation setup is thus given by $2 \times N_{\text{muscles}} + 1$.

4.4.2 Reward Design

Proper design of reward functions and reward shaping is essential for reinforcement learning algorithms to learn the desired behavior through optimization successfully. A well-designed reward should balance encouraging desired behavior and penalizing failure scenarios while aligning with long-term objectives, such as maintaining steady stepping motions for all future cycles. Furthermore, the reward should be carefully tuned to avoid discouraging agent exploration. Our reward design incorporates three components: model sustaining, motion imitation, and alignment to the specific goal.

Model Sustaining: To encourage the agent to learn a policy that sustains the model for a longer time, we provide a constant but small amount of alive reward for every time step, denoted as r_{alive}. Conversely, we apply a significant penalty when the model falls, teaching the agent to avoid actions that lead to instability or falls, denoted as c_{fall}. In practice, we adjust the amount of this penalty based on the remaining time for the episode:

$$c_{\text{fall}}(t) = w_{\text{fall}} \times \frac{t_{\max} - t}{t_{\max}}, \tag{4.18}$$

where w_{fall} controls the maximum amount of falling penalty.

Furthermore, a moderate amount of step reward, denoted as r_{step}, is given for the model making reasonable steps. The step reward is given only when the ground contact with the feet is detected, and the contact is from a different foot than the last time the step reward was emitted. This helps to encourage the model to alternate between feet when stepping, promoting a more natural gait. This design is derived from one of the examples in (Kidziński et al., 2018).

Overall, the model sustaining reward component can be summarized as follows:

$$r_{\text{sustain}}(t, \boldsymbol{s}) = \begin{cases} -c_{\text{fall}}(t) & \text{if the model falls;} \\ r_{\text{step}} + r_{\text{alive}} & \text{if a new step is detected;} \\ r_{\text{alive}} & \text{otherwise.} \end{cases} \tag{4.19}$$

Typically, the scales of the aforementioned rewards are set at $r_{\text{alive}} = 0.01$, $r_{\text{step}} = 3.0$, and $w_{\text{fall}} = 10.0$. These values were selected through trials to provide a good balance between encouraging the model to remain alive and take natural steps while strongly discouraging falling.

Motion Imitation: Despite the ability to generate long-lasting motions with stepping-like behaviors, using solely the model sustaining reward often results in unnatural locomotion patterns, even when combined with the rhythmic patterns from CPG. The learned motions exhibit biomechanically implausible movements and a lack of naturalness. To address this limitation, we draw inspiration from the reward design in the DeepMimic method (Peng et al., 2018) and the Trajectory Mimicking idea introduced in (Lee et al., 2019), and introduce a mimic reward term that encourages the agent to imitate reference motion data as follows:

$$r_{\text{mimic}}(t, \boldsymbol{s}) = w^q_{\text{mimic}} r^q_{\text{mimic}}(t, \boldsymbol{s}) + w^e_{\text{mimic}} r^e_{\text{mimic}}(t, \boldsymbol{s}), \tag{4.20}$$

where r^q_{mimic} and r^e_{mimic} refer to the reward terms that promote the similarity of the joint angles and the positions of the end-effectors, respectively, and w^q_{mimic} and w^e_{mimic} are their corresponding weights.

The individual reward terms are calculated as

$$r^{q}_{\text{mimic}}(t, \boldsymbol{s}) = \exp\left[-\sigma_q \sum_j \|\hat{q}_j(t) - q_j(\boldsymbol{s})\|^2\right], \tag{4.21}$$

$$r^{e}_{\text{mimic}}(t, \boldsymbol{s}) = \exp\left[-\sigma_e \sum_e \|\hat{\boldsymbol{p}}_e(t) - \boldsymbol{p}_e(\boldsymbol{s})\|^2\right], \tag{4.22}$$

where j and e are the indices of the joints and the end-effectors, $\hat{q}_j(t)$ and $\hat{\boldsymbol{p}}_e(t)$ denote the reference joint angles and end-effector positions at time t, respectively, while $q_j(\boldsymbol{s})$ and $\boldsymbol{p}_e(\boldsymbol{s})$ represent the corresponding values of the agent in state $\boldsymbol{s}$. The parameters σ_q and σ_e control the sensitivity of the reward terms to the differences between the reference and agent values, which are typically set to 3.0 and 15.0 in our experiments.

Compared to the reward term in DeepMimic (Peng et al., 2018), our mimic reward differs in two aspects. First, ours does not include a term for the similarity of joint velocities. In practice, we found this metric can be unstable, which may arise from the sensitivity of velocity measurements in the reference motions we use. Moreover, unlike both (Peng et al., 2018) and (Lee et al., 2019), which use the quaternion difference for the joint orientation in r^{q}_{mimic}, we opt for a simpler approach and directly take the norm of the difference between the joint angles. Our experiments show that this simplified configuration can work effectively with reduced computational complexity. The mimic reward term complements the sustaining reward by providing explicit guidance towards desired natural bipedal motions.

Goal Alignment: We use a goal alignment reward to encourage the agent to learn to perform locomotion control in a particular style while maintaining imitation of the reference motions. Our goal representation is a 4-dimensional vector that acts as part of the input to the two networks in the policy architecture. The first two dimensions represent the desired direction of motion as a normalized vector in the x-y plane, while the next two scalar values represent the desired velocity and pelvis height, respectively.

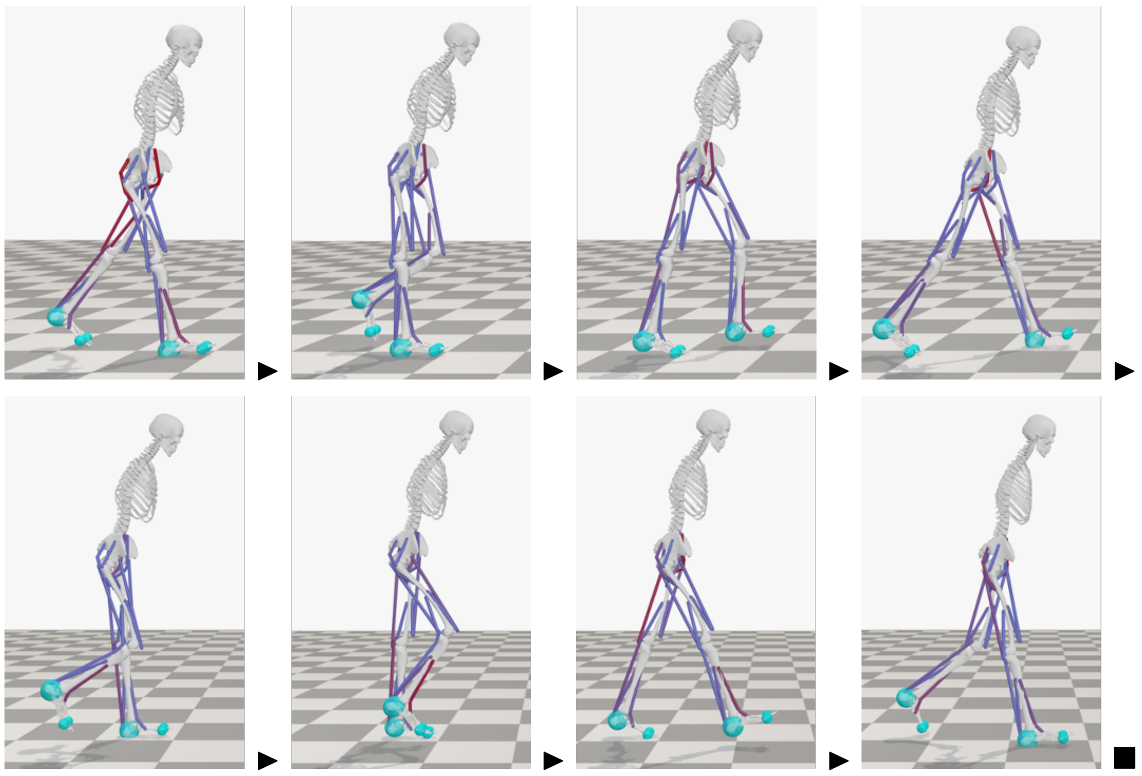


Figure 4.5: A sequence of snapshots demonstrating a complete walking forward cycle generated by our trained locomotion controller.

The goal reward may be formalized as

$$r_{\text{goal}}(t, \boldsymbol{s}) = w^d_{\text{goal}} \|\hat{\boldsymbol{d}} - \boldsymbol{d}(\boldsymbol{s})\|^2 + w^v_{\text{goal}} \|\hat{v} - v(\boldsymbol{s})\|^2 + w^h_{\text{goal}} \|\hat{h} - h(\boldsymbol{s})\|^2, \tag{4.23}$$

where w^d_{goal}, w^v_{goal}, and w^h_{goal} are the weights for the rewards that promote the alignment to the desired direction, speed, and pelvis height, respectively, and $\hat{\boldsymbol{d}}$, $\hat{v}$, and $\hat{h}$ denote the target direction vector, speed, and pelvis height provided in the goal vector, while $\boldsymbol{d}(\boldsymbol{s})$, $v(\boldsymbol{s})$, and $h(\boldsymbol{s})$ represent the corresponding values of the agent in state $\boldsymbol{s}$.

4.5 Results

In this section, we demonstrate and analyze the results from our experiments. We will compare the result of our proposed locomotion controller, integrated with the CPG and dual-modular architecture, with two established methods, namely the CPG-based

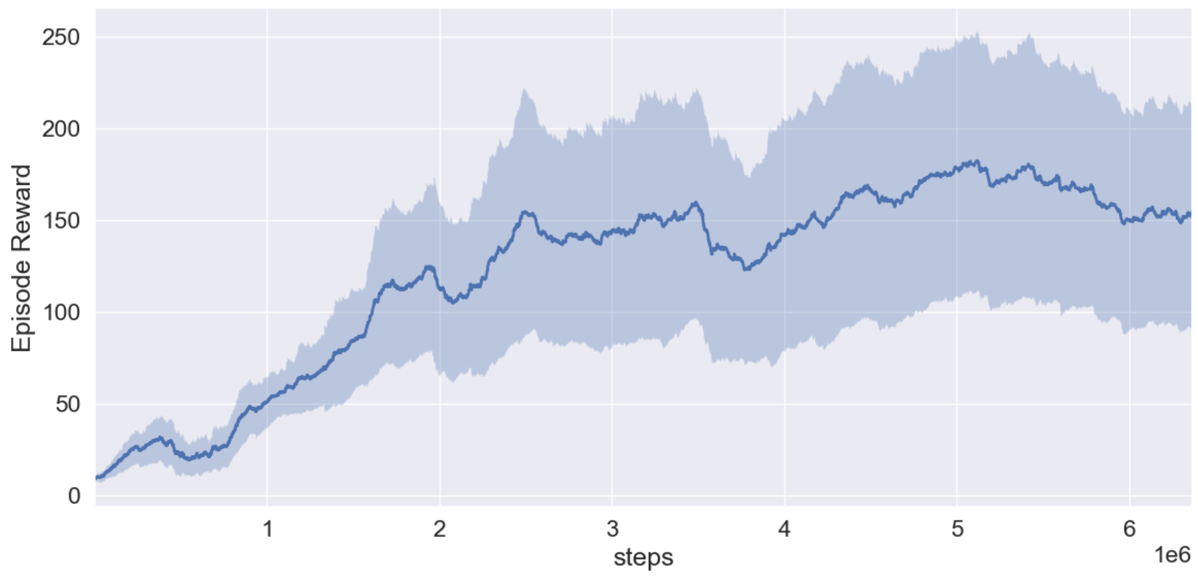


Figure 4.6: The learning curve for the proposed locomotion controller, showing the improvement in cumulative rewards. The solid line represents the rolling mean, and the shaded area represents the rolling standard deviation. The learning procedure terminates at 6.5 million steps, where stable walking forward behavior is achieved.

swimming control scheme (Si et al., 2014) and the feedback-based DeepMimic method (Peng et al., 2018). We also conducted several ablation experiments to further validate the benefits of the dual-modular design and CPG tuning mechanism.

4.5.1 Training

As mentioned in Section 4.3, we applied both the SAC and PPO methods to our biomechanical model. While PPO is comparatively stable, the advantage of sample efficiency for SAC makes it outperform most of the experiments. Figure 4.6 demonstrates our training procedure and learning curve of the proposed locomotion control model with dual-modular architecture, where we can achieve the desired performance with 6 million steps.

The training was conducted on a workstation equipped with an Intel Core 14th Gen i7-14700KF CPU, 32GB of RAM, and an NVIDIA GeForce RTX 4070 GPU with 12GB of VRAM. The simulation environment was built using OpenSim 4.1, and the reinforcement learning algorithms were implemented using the PyTorch library and the Stable-Baselines3

(Raffin et al., 2021) framework. The entire training process took approximately 32 hours to complete.

4.5.2 Walking Forward

One of the fundamental tasks for evaluating the performance of our locomotion controller is its ability to generate stable and natural forward walking motions. Figure 4.5 illustrates a sequence of snapshots where our control framework generates muscle activations to drive the biomechanical human model to walk forward continuously in the OpenSim simulation environment. The upper body maintains a slightly forward-leaning posture throughout the walking cycle, contributing to the overall balance and stability of the motion.

The successful generation of this natural walking behavior demonstrates the effectiveness of our CPG-based controller, where the learned rhythmic patterns from the CPG module produce coordinated muscle activations as the basis for locomotion. The dual-modular architecture, consisting of the CPG Tuner and Reflex Controller, allows the model to adjust its gait to the environment and maintain a steady forward pace.

4.5.3 Comparison: Dual-Modular versus Flattened Architecture

To validate the benefits of a dual-modular architecture, we compare our proposed controller's performance with a variant using a flattened architecture, where the CPG Tuner and Reflex Controller modules are united into a single fully-connected network. In particular, the input signals are simplified as a concatenation of the body state $\boldsymbol{s}$, CPG module status $\boldsymbol{a}_{\text{cpg}}$ and Φ, and the goal representation $\boldsymbol{g}$, while the output signals combine both CPG modulation signals and the reflex control adjustments to muscle activations. To ensure a fair comparison, the number of trainable parameters is set to a similar amount compared to the dual-modular version.

Figure 4.7 presents the reinforcement learning curves for both architectures, illustrating the agents' progress during training. It is evident that our dual-modular architecture demonstrates faster learning and better performance in terms of the highest episode

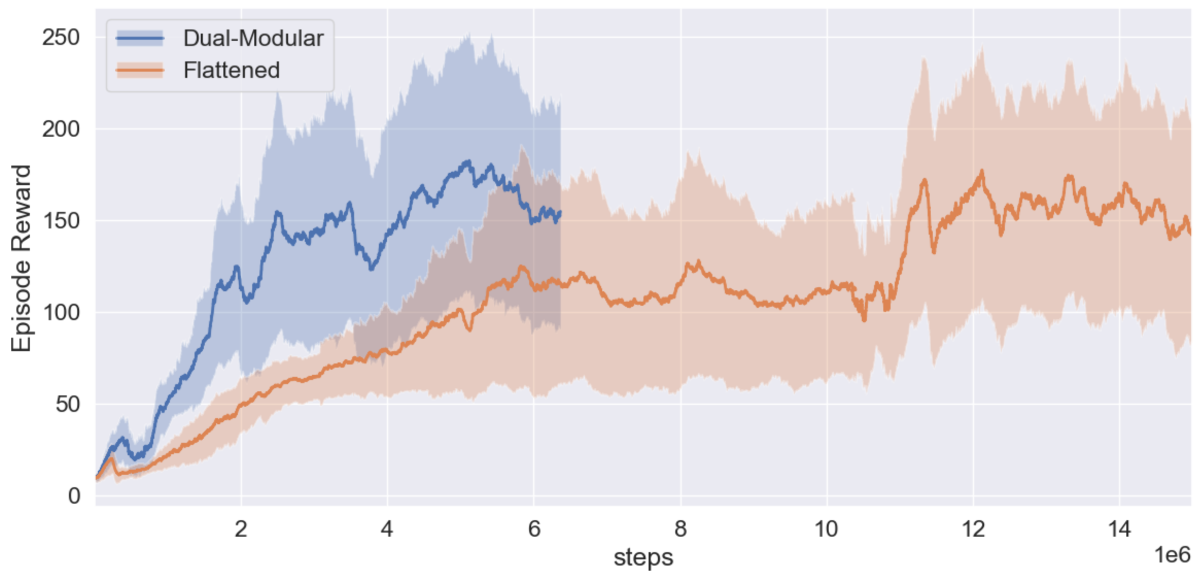


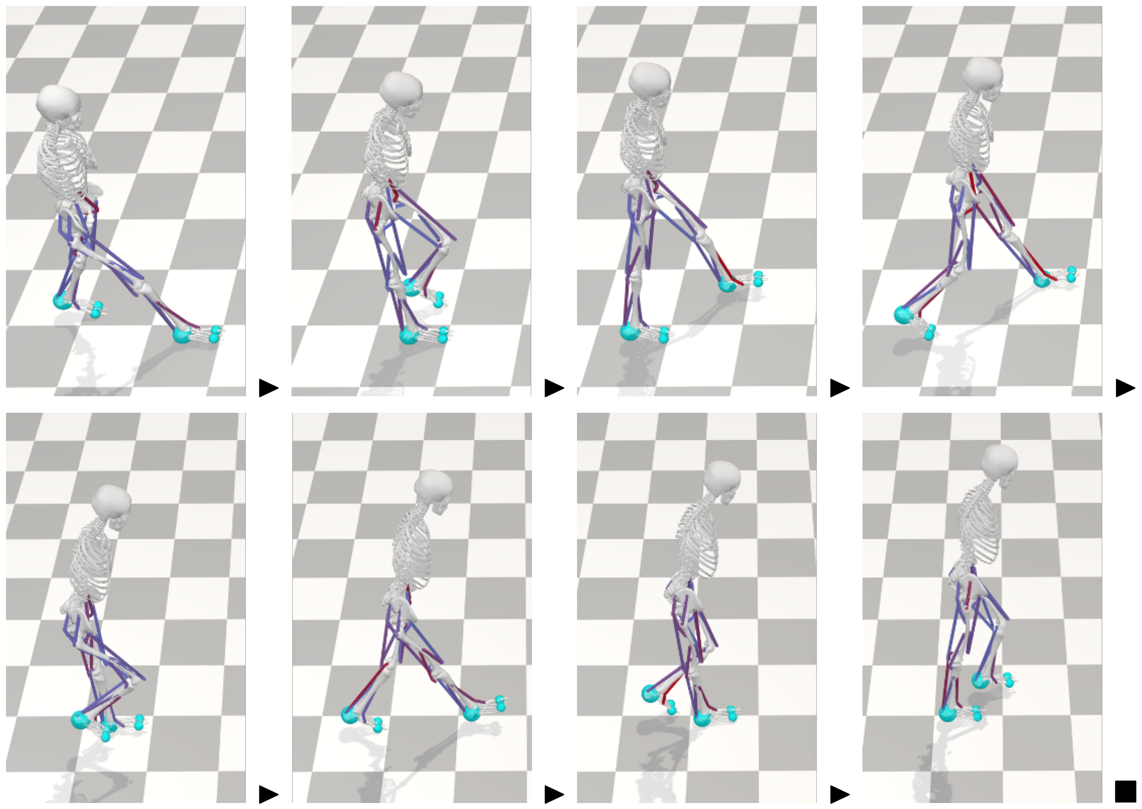
Figure 4.7: Comparison of the learning curves for the dual-modular and flattened architectures. The dual-modular architecture demonstrates faster learning and higher best performance compared to the flattened architecture.

reward compared to the flattened architecture. This advantage can be attributed to the separation of the CPG Tuner and Reflex Controller modules, which allows each module to focus on its specific role in the control process with the relevant input signals. The CPG Tuner module learns to modulate CPG patterns for base locomotion, while the Reflex Controller module learns to provide corrective muscle activation adjustments based on the current model state and environment.

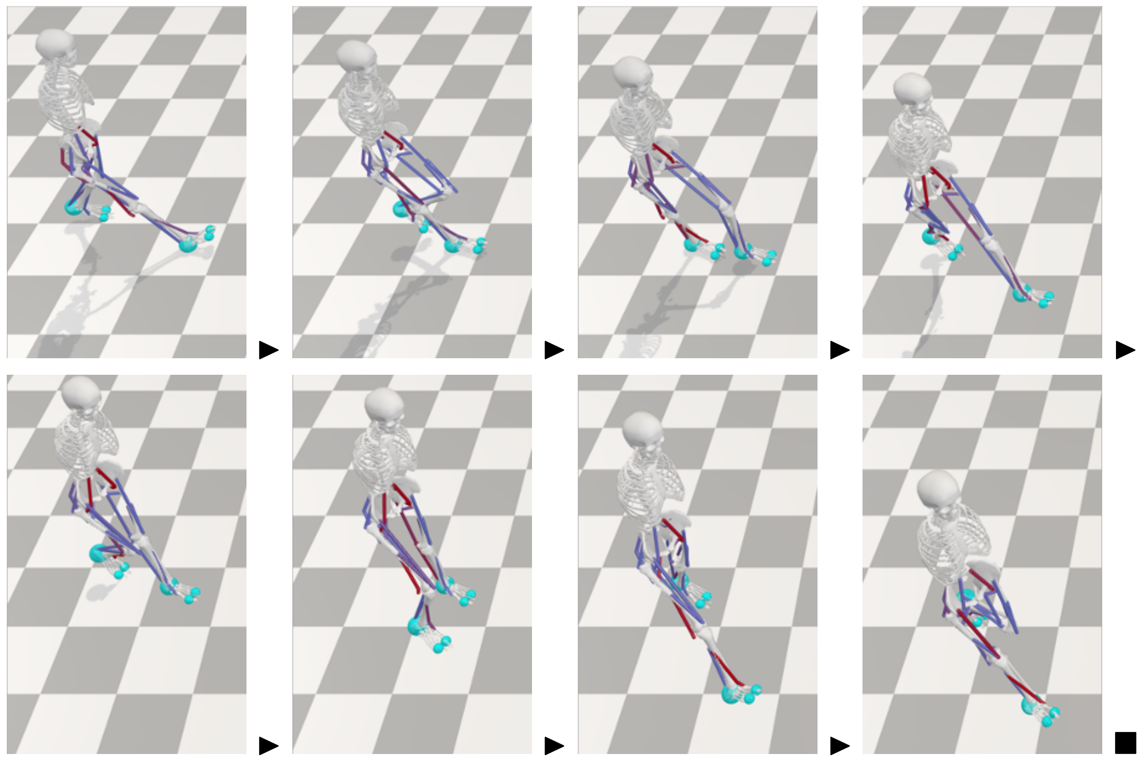
4.5.4 Goal-Based Control

Figure 4.8 demonstrates the controller's ability to generate walking motions that turn left or right based on a goal direction specified as a normalized vector in the x-y plane as part of the goal representation. The goal is incorporated into the controller's architecture as part of the input to the CPG Tuner and Reflex Controller modules, allowing the controller to adapt its behavior to achieve the desired turning motion.

To generate the turning motion, the controller adjusts the gait pattern by modulating the CPG signals and generating appropriate muscle activation adjustments. The inner leg



(a) Turning left.



(b) Turing right.

Figure 4.8: Sequence of snapshots demonstrating the model walking in a left or right turn based on a specified target direction.

(the leg closer to the center of the turn) takes shorter steps and has a longer stance phase, while the outer leg takes longer steps and has a shorter stance phase. This asymmetry in the gait pattern allows the model to smoothly change direction while maintaining forward progression.

Throughout the turning motion, the controller ensures that the model maintains a stable pace while minimizing deviations from the reference motions, where the Reflex Controller module plays a crucial role. The turning motion demonstrates the adaptability and versatility of our controller.

CHAPTER 5

Learning-Based Facial Expression Transfer From Images and Videos

5.1 Overview

In this chapter, we introduce an innovative learning-based method for transferring facial expressions from images and videos to the biomechanical model of the face-head-neck complex as discussed in Section 3.3. We utilize the Facial Action Coding System (FACS) as an intermediate representation of facial expressions and train a deep neural network to map FACS Action Units (AUs) to facial muscle activation signals. This chapter is a reproduction of our published work (Zeng et al., 2021b) with additional data and technical details not included in the original publication due to length limitations.

5.2 Facial Action Coding System

The Facial Action Coding System (FACS) is a comprehensive system for taxonomizing human facial movements based on the underlying muscle anatomy. Developed by Paul Ekman and Wallace V. Friesen in 1978 (Ekman and Friesen, 1978), the FACS has become a standard tool in facial expression analysis and synthesis. The system defines a set of Action Units (AUs), each representing the movement or activity of a specific facial muscle or muscle group. The original FACS scheme introduced 44 AUs for facial surface movements, 6 AUs for eye gaze, and 8 AUs for head movements (Wojdeł and Rothkrantz, 2005). These AUs can be combined to represent a wide range of human facial expressions, from subtle to more complex ones, and they have also been adapted for analyzing facial expressions in non-human species (Parr et al., 2007).

Many works about facial expression synthesis and computer animation have employed FACS to generate realistic facial expressions with virtual characters. For example, Platt and Badler (1981) and Wojdeł and Rothkrantz (2005) showcase model generation techniques using FACS AUs as the basis for their animation control parameters. More recently, researchers have developed methods to map FACS AUs to facial mesh deformations automatically (Sumathi et al., 2012), with the help of advanced machine learning techniques.

In our work, we utilize OpenFace 2.0 (Baltrušaitis et al., 2018) for facial expression recognition and head pose estimation. OpenFace is an open-source tool that provides state-of-the-art facial behavior analysis, including facial landmark detection, head pose estimation, and facial action unit recognition based on FACS. It employs a combination of computer vision and machine learning techniques, such as Convolutional Experts Constrained Local Models (CE-CLMs) for facial landmark detection and tracking. Specifically, OpenFace uses CE-CLMs to detect and track 68 facial landmarks in real-time, providing a robust and accurate foundation for further analysis. In the context of our use case, OpenFace provides a pre-trained deep neural network that takes the aligned facial image as input and outputs the intensity estimates for 18 FACS AUs that contribute to the expression representation, as illustrated in Table 5.1.

5.3 Training Data Generation

In this section, we introduce the training data generation pipeline for our facial expression transfer system, which utilizes the OpenFace 2.0 toolchain described in the previous section.

The overall process of generating training data comprises two principal steps: first, a substantial number of facial expressions are synthesized by randomly combining different muscle activation patterns on our biomechanical model; second, OpenFace is utilized to extract the corresponding FACS AU intensities from the rendered images of these

AU	Full Name	Illustration	Intensity	Presence
AU1	Inner brow raiser		✓	✓
AU2	Outer brow raiser		✓	✓
AU4	Brow lowerer		✓	✓
AU5	Upper lid raiser		✓	✓
AU6	Cheek raiser		✓	✓
AU7	Lid tightener		✓	✓
AU9	Nose wrinkler		✓	✓
AU10	Upper lip raiser		✓	✓
AU12	Lip corner puller		✓	✓
AU14	Dimpler		✓	✓
AU15	Lip corner depressor		✓	✓
AU17	Chin raiser		✓	✓
AU20	Lip stretched		✓	✓
AU23	Lip tightener		✓	✓
AU25	Lips part		✓	✓
AU26	Jaw drop		✓	✓
AU28	Lip suck			✓
AU45	Blink		✓	✓

Table 5.1: List of 18 FACS AUs used in OpenFace 2.0. According to (Baltrušaitis et al., 2018), both the intensity and presence of these AUs are predicted, except for AU28, for which only the presence are predicted. This table is adapted and extended from the OpenFace paper.

synthesized expressions.

5.3.1 Muscle-Based Facial Expression Synthesis

To generate a variety of facial expressions, we first define a set of basic muscle activation patterns for basic expressions. Given n muscles contributing to the facial soft tissue simulation ($n = 52$ for our model), we define W_e as a set of weights $w_{e,i}$, where $w_{e,i}$ represents the effect of the i-th muscle ($1 \leq i \leq n$) for the basic expression e. The activation a_i for each muscle in a synthesized expression is then calculated as the weighted sum of the weights of the basic expressions, given by $a_i = \sum_e w_{e,i} s_e$, where s_e is the scaling term that determines the intensity of each basic expression.

We manually fine-tune the weights for a single basic expression by visually analyzing the synthesized expression along with expert knowledge derived from (Lee and Terzopoulos, 2006). As a result, we constructed the muscle weights over six basic expressions, namely (1) Joy, (2) Sadness, (3) Anger, (4) Fear, (5) Disgust, and (6) Surprise. The corresponding sets are W_{Joy}, W_{Sadness}, W_{Anger}, W_{Fear}, W_{Disgust}, and W_{Surprise}, respectively.

We then sample the scaling terms s_e independently for each base expression from a uniform distribution in the range $[0.0, 1.0]$, thus constructing a randomized expression. To further enrich the diversity of the dataset, we assign a random value to the jaw rotation parameter while keeping the jaw twist and jaw slide values at 0.5, which corresponds to a neutral jaw position. By repeating this process, we collect a set A of muscle activations representing a wide range of randomized facial expressions.

5.3.2 Action Units Recognition and Labeling

The muscle activation set A, generated in the previous step, is employed to synthesize a vast array of facial expression images with our biomechanical model. For each rendering, a random set of muscle activations from A is selected and applied to the model. In order to enhance the diversity of the dataset, we also randomize the neck poses and face orientations by applying random rotations to the head, which are obtained by randomly sampling angles for each axis of rotation (pitch, yaw, and roll).

Subsequently, OpenFace 2.0 is employed to label the rendered images with FACS AUs. For each image, OpenFace detects the facial landmarks, estimates the head pose, and predicts the intensities of 18 FACS AUs, as mentioned in previous sections and Table 5.1. However, one of the predicted AUs (AU #28) does not contain intensity information, so it is excluded from our labeling. Consequently, each rendered facial expression image is paired with a set of 17 AU intensities, which serve as labels for the corresponding muscle activations used to generate the expression.

5.4 Model Architecture and Training

In this section, we present the architecture of our deep learning model for facial expression transfer and provide details on the training process. Figure 5.1 shows an overview of the proposed transfer framework as well as the controller modules over the offline training and online transferring procedures.

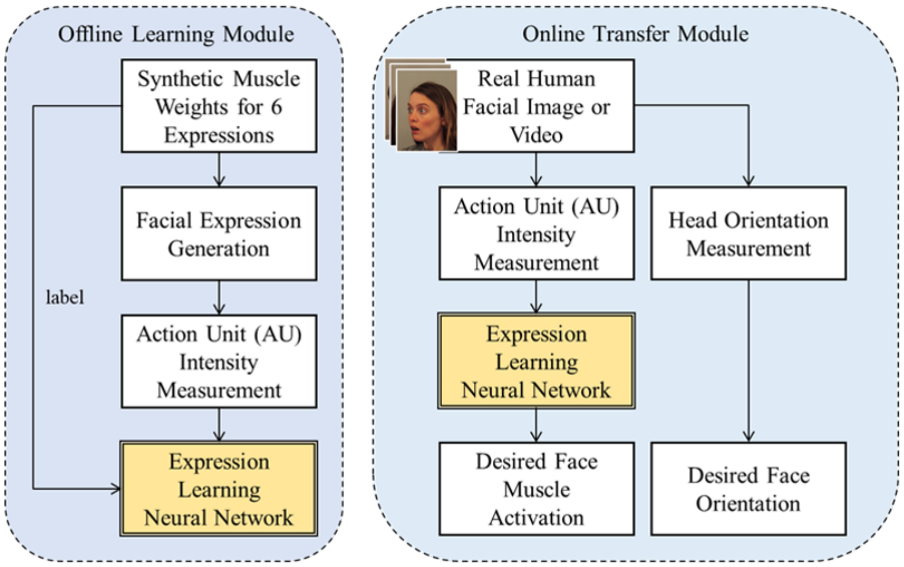
5.4.1 Network Architecture

Our Expression Learning Neural Network (Figure 5.1a) takes FACS AUs extracted from facial imagery data by OpenFace as input features. These AUs serve as an encoding of the facial expression, which the network then uses to generate facial muscle activation signals for our biomechanical model to reproduce the expression.

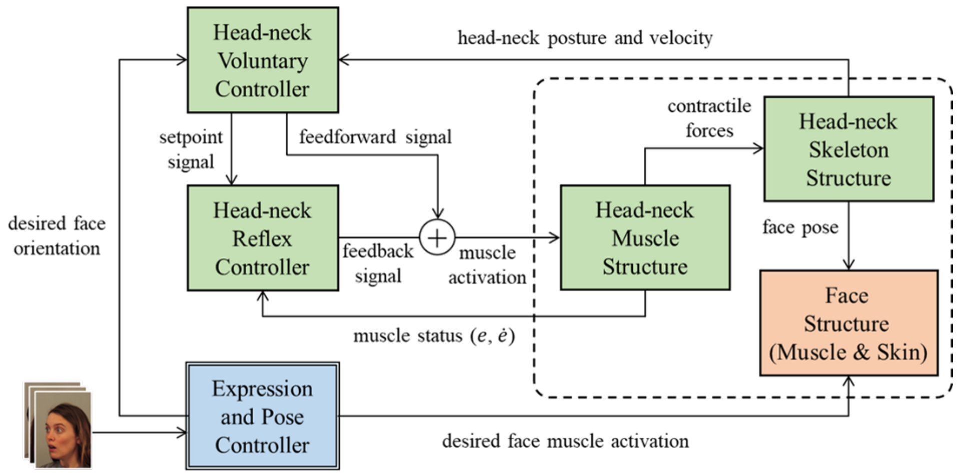
Figure 5.2 illustrates the fully-connected architecture of our network. The input layer comprises 17 neurons, each receiving a normalized AU intensity value. The network comprises four fully-connected hidden layers, each containing 100 neurons with ReLU activation functions. The output layer generates 56 signals, with 52 signals representing the activations a_i $(1 \leq i \leq 52)$ for the 26 pairs of facial muscles and the remaining 4 signals representing jaw rotation, jaw slide, jaw twist, and an auxiliary value. The network, implemented using Keras with a TensorFlow backend, has a total of 37,300 learnable weights.

5.4.2 Network Training

To train our Expression Learning Neural Network, we use the Adam optimizer (Kingma and Ba, 2014) with a Mean Squared Error (MSE) loss function. We set the batch size to 32 and the learning rate to 0.01, based on empirical experiments and considering the trade-off between convergence speed and stability. The network is trained for 100 epochs


Offline Learning Module
Synthetic Muscle Weights for 6 Expressions
Facial Expression Generation
Action Unit (AU) Intensity Measurement
Expression Learning Neural Network
label
Online Transfer Module
Real Human Facial Image or Video
Action Unit (AU) Intensity Measurement
Head Orientation Measurement
Expression Learning Neural Network
Desired Face Muscle Activation
Desired Face Orientation

(a) Expression Transfer Framework


Head-neck Voluntary Controller
head-neck posture and velocity
contractile forces
Head-neck Skeleton Structure
setpoint signal
feedforward signal
Head-neck Reflex Controller
feedback signal
muscle activation
Head-neck Muscle Structure
face pose
desired face orientation
Face Structure (Muscle & Skin)
muscle status $(e, \dot{e})$
Expression and Pose Controller
desired face muscle activation

(b) Hierarchy of Controllers & Flow of Signals

Figure 5.1: Overview of the proposed facial expression transfer framework (a) and the controllers for the musculoskeletal and facial soft-tissue structures (b). The Expression Learning Neural Network (yellow) is trained offline before being integrated into the online transfer pipeline. During the transfer task, an input facial image or video sequence is processed to generate the desired facial muscle activations and head orientation. Subsequently, the control model utilizes these outputs to simulate the biomechanical face model (orange), which reproduces the corresponding facial expression, and to drive the head-neck musculoskeletal model (green), which produces the desired head pose. From (Zeng et al., 2021b)

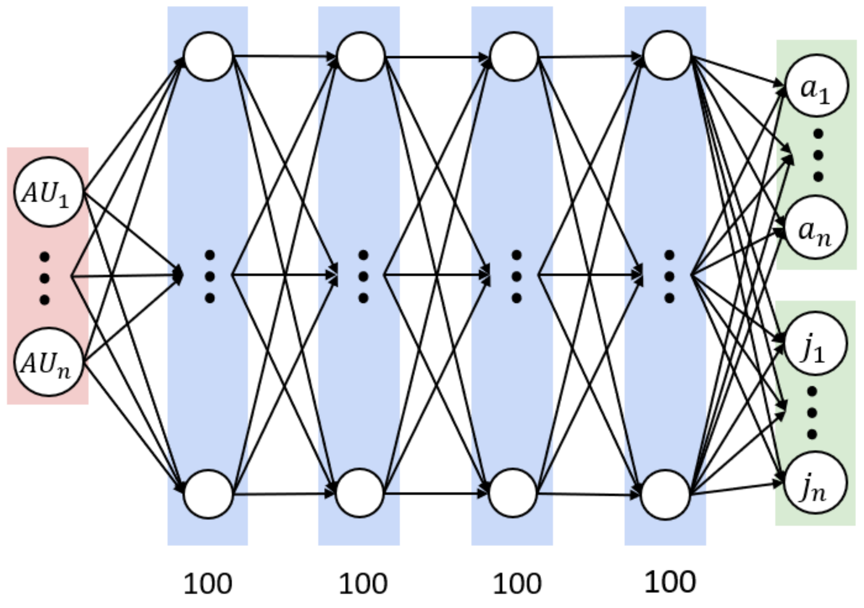


Figure 5.2: Architecture of our Expression Learning Neural Network.

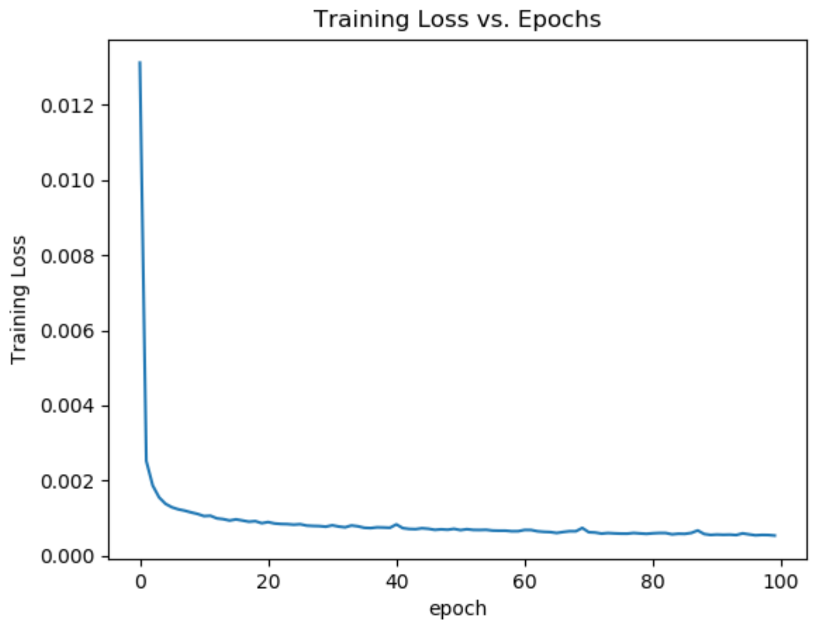


Figure 5.3: Convergence of the MSE loss during the training of our Expression Learning Neural Network over 100 epochs.

on an NVIDIA GeForce GTX 1650 GPU installed on a Windows 10 machine with a 2.6GHz Intel Core i7-9750H CPU. Figure 5.3 shows the convergence of the training loss over the epochs.

To address the discrepancy in AU intensity values between the synthetic training data and the real faces encountered during the expression transfer task, we normalize the intensity values of each AU class across all training pairs. This normalization step ensures consistency and enhances the generalizability of the trained network. The training dataset comprises a total of 6,000 pairs, with approximately 1,000 pairs for each of the six basic expressions (joy, sadness, anger, fear, disgust, and surprise).

5.5 Experiments and Results

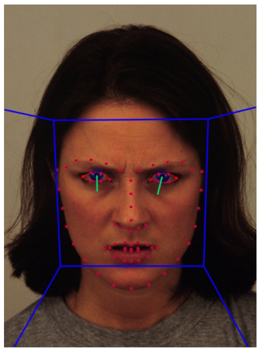
In this section, we present the experimental setup and results of our facial expression transfer method. We evaluated the effectiveness of our approach using several publicly available datasets, including the Karolinska Directed Emotional Faces (KDEF) dataset (Calvo and Lundqvist, 2008), the Cohn-Kanade (CK) dataset (Kanade et al., 2000), and the Extended Cohn-Kanade (CK+) dataset (Lucey et al., 2010). Appendix A presents detailed information about these datasets.

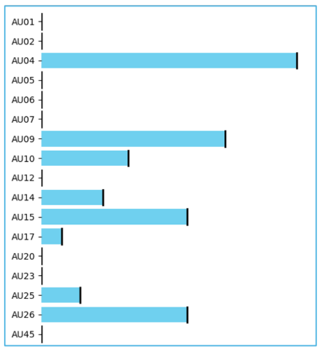
5.5.1 Online Facial Expression Transfer

With the trained model, we establish a mapping between the FACS AUs and the muscle activations required to reproduce the corresponding facial expressions. This enables us to perform online facial expression transfer using the AU intensities extracted by OpenFace from input facial images or videos. As demonstrated in Figure 5.4, the whole pipeline can reconstruct the facial expression from arbitrary face imagery, provided that OpenFace can accurately detect the AUs. For video sequences, we process each frame independently and generate a corresponding sequence of muscle activations to animate the model.

We evaluated the performance of our facial expression transfer method on a wide range

(a) Input Image


(b) Face Recognition



(c) Output AUs

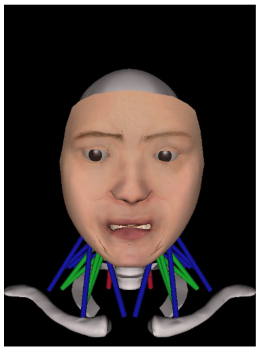
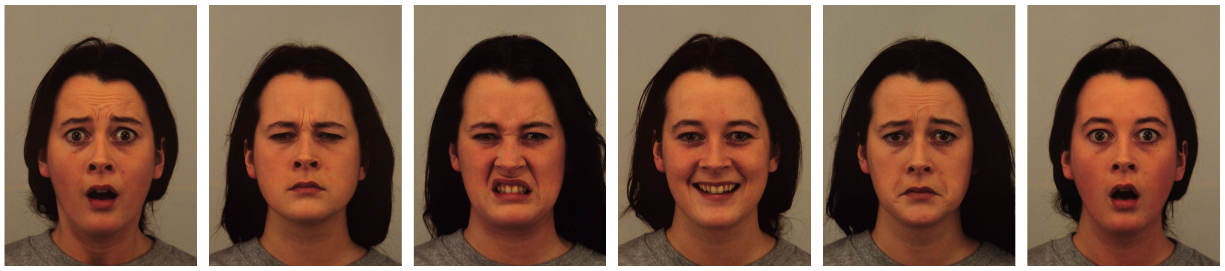

(d) Reconstructed

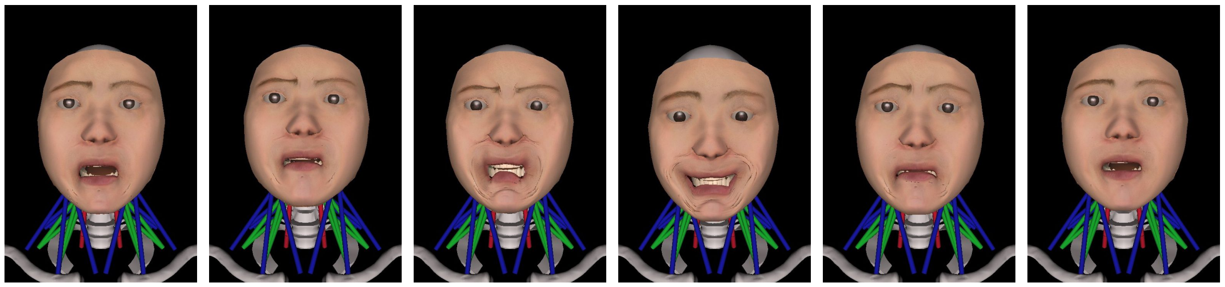
Figure 5.4: The procedure of online facial expression transfer. Given an input facial image or video frame (a), OpenFace performs facial landmark recognition (b) and detects the AU intensities (c), which are then fed into our trained Expression Learning Neural Network. The network generates the appropriate muscle activations, which are used to drive our biomechanical facial model to reproduce the observed expression (d).

of facial expressions from the KDEF dataset. Figure 5.5 presents the transfer results for each of the six basic expressions performed by two subjects (one female and one male). We can observe that our approach successfully reproduces the reference expressions and captures the subtle facial features that distinguish each individual's nuanced expression characteristics. For instance, the female subject displays a more intense expression of joy compared to the male subject, as evidenced by the wider smile and more pronounced cheek raising. Similarly, the female subject's expression of fear is characterized by a stronger jaw drop compared to the male subject.

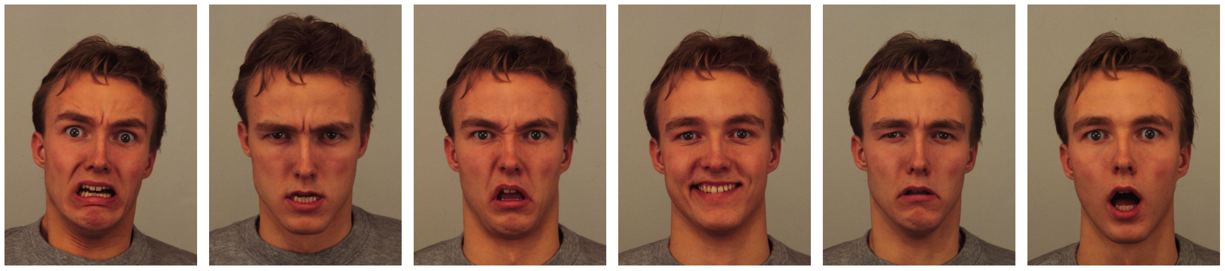
We also investigated the impact of AU normalization and jaw activation on the quality of the transferred expressions, as demonstrated in Figure 5.6. By comparing the transfer result with and without AU normalization mentioned in Section 5.4.2, we can observe that the normalization effectively helps the pipeline to reduce artifacts and over-exaggerated facial features. in a later section, we will also present a quantitative comparison indicating that AU normalization improves the transfer accuracy. Meanwhile, enabling jaw activation leads to more accurate and lifelike reproductions, particularly for those that involve significant mouth movements, such as surprise and fear.



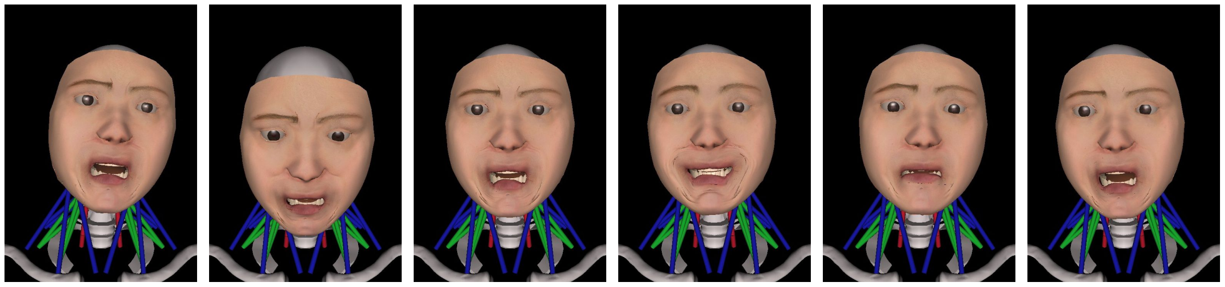
(a) Input images of Subject 4 in KDEF



(b) Reconstructed results of Subject 4 in KDEF

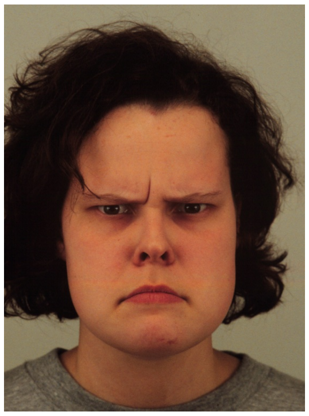


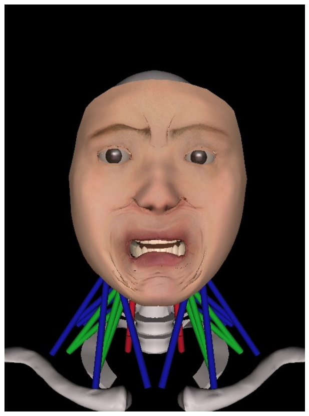
(c) Input images of Subject 5 in KDEF

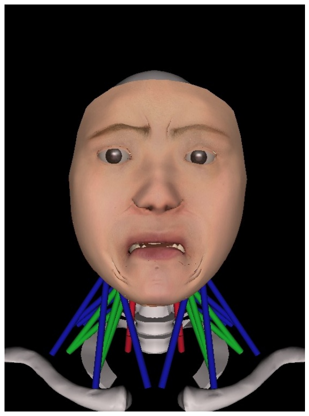


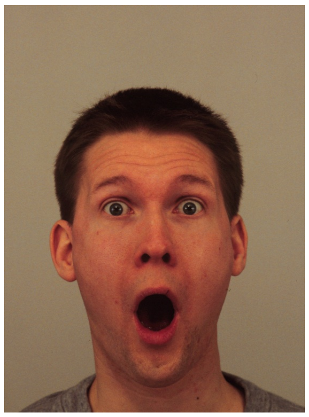
(d) Reconstructed results of Subject 5 in KDEF

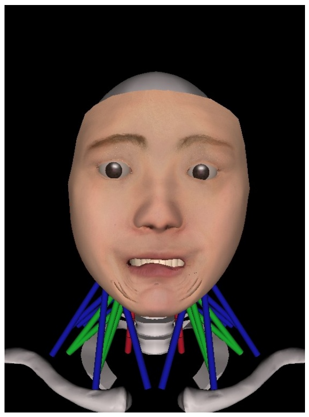
Figure 5.5: Facial expression transfer results for six basic expressions performed by two subjects from the KDEF dataset. The expressions are fear, anger, disgust, joy, sadness, and surprise, respectively.


(a) Input Image


(b) No Normalization


(c) With Normalization


(d) Input Image


(e) No Jaw Activation

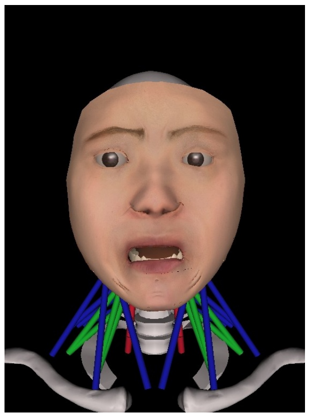
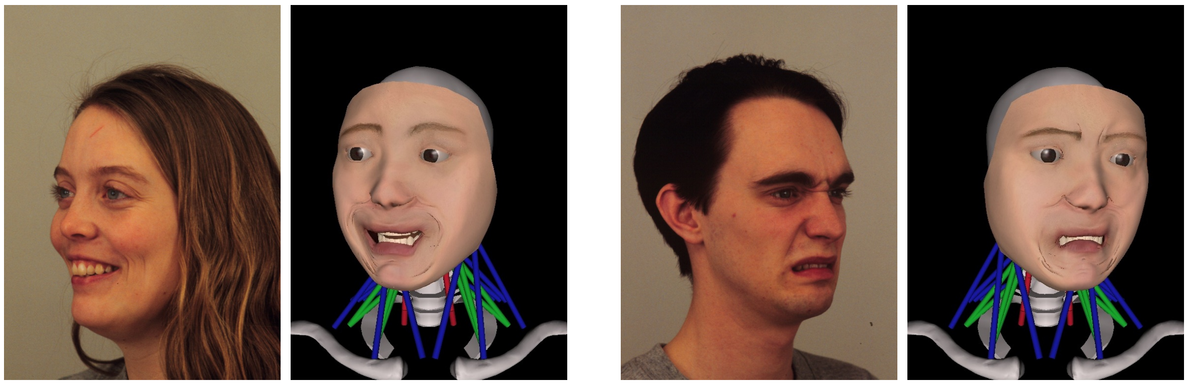

(f) With Jaw Activation

Figure 5.6: Comparison of transfer results over AU normalization and jaw activation. In the first row, we compare the result with and without AU normalization, and the second row shows the result with jaw activation enabled and disabled.

5.5.2 Head Orientation Transfer

In addition to facial expressions, our method also enables the transfer of head orientation from the input images or videos to the biomechanical model. Figure 5.7 presents examples of head pose transfer along with facial expressions for different subjects from the KDEF dataset.

The head orientation in our biomechanical model is controlled at the musculoskeletal level by a reflex controller, where muscle activations around the neck are adjusted to achieve the desired angle. Further details about the reflex controller and its role in tuning head orientation are presented by Lee and Terzopoulos (2006).



(a) Subject 1 (b) Subject 2

Figure 5.7: Examples transferring both head orientation and facial expression for two subjects in KDEF dataset.

	KDEF (unnormalized)	KDEF (normalized)	RAVDESS (video)	CK (video)
MSE_{AU1}	0.191	0.118	0.012	0.038
MSE_{AU4}	0.088	0.064	0.222	0.054
MSE_{AU6}	0.186	0.186	0.107	0.059
MSE_{AU9}	0.133	0.068	0.044	0.015
MSE_{AU12}	0.011	0.028	0.083	0.008
MSE_{AU15}	0.155	0.109	0.016	0.023
MSE_{AU20}	0.129	0.146	0.077	0.057
MSE_{AU25}	0.092	0.052	0.036	0.034
Average	0.157	0.129	0.086	0.048

Table 5.2: Average Mean Squared Error (MSE) of selected Action Units (AUs) over the transfer results and their corresponding references in different datasets. For the KDEF dataset, we also compare the results with and without the AU normalization.

5.5.3 Quantitative Accuracy Evaluation

To quantitatively evaluate the accuracy of our facial expression transfer method, we compute the Mean Squared Error (MSE) between the OpenFace recognition results over the input and reconstructed faces. Table 5.2 shows the average MSE of selected AUs over the transfer results and their corresponding reference in different datasets. We selected 8 of the 17 AUs that served as the input to our Expression Learning Neural Network.

From the results, we can observe that our method achieves low MSE values across different datasets, indicating the accuracy of our facial expression transfer. The average

MSE ranges from 0.048 in the CK dataset to 0.157 in the unnormalized KDEF dataset. The lower MSE values in the video datasets (RAVDESS and CK) compared to the KDEF dataset can be attributed to the more subtle and continuous changes in facial expressions present in videos, which our method can effectively capture and transfer.

Comparing the results between the unnormalized and normalized KDEF datasets, we can see that the AU normalization leads to a notable improvement in the transfer accuracy. The average MSE decreases from 0.157 in the unnormalized dataset to 0.129 in the normalized dataset. This improvement is consistent across most of the selected AUs, demonstrating the effectiveness of AU normalization. It is also worth noting that the MSE values for individual AUs vary depending on the dataset and the specific AU. For example, AU6 (cheek raiser) has relatively high MSE values in the KDEF and RAVDESS datasets, while AU12 (lip corner puller) has low MSE values in all datasets.

CHAPTER 6

Conclusions and Future Work

6.1 Conclusions

This dissertation primarily examined two significant areas in the field of human modeling and animation: the locomotion control of muscle-actuated biomechanical body models and anatomically accurate human facial expression transfer from images and videos to a model of the face-head-neck biomechanical complex. The long-term motivation behind this research is to create natural, biomimetic virtual characters by addressing the limitations of traditional motion capture-based approaches, which often lack adaptability and responsiveness.

A methodology that carries through our endeavor is the integration of prior knowledge into the machine learning framework. While end-to-end learning approaches have gained popularity in recent years due to increasing computational power and their ability to learn complex patterns from vast amounts of data, we argue that incorporating domain knowledge and biological principles remains valuable in the context of biomimetic modeling and control. By leveraging expert insights in human anatomy, physiology, and neuromuscular control, we can guide the learning process toward more plausible and interpretable models.

In our work on locomotion control, we proposed a synergistic combination of Central Pattern Generators (CPGs) and deep reinforcement learning to achieve adaptive and natural locomotion control in a biomechanical virtual human model. The CPG provides a biologically-inspired means for generating rhythmic muscle activation patterns, while the reinforcement learning component enables the model to adapt to the environment. We demonstrated the effectiveness of our approach through extensive experiments, showing that the model can learn stable walking gaits and adapt to different goals. The incorporation of domain knowledge in the form of CPGs proved to be a valuable strategy.

Furthermore, the dual-module architecture, consisting of a CPG Tuner and a Reflex Controller, provided a clear separation of responsibilities and improved learning speed and performance compared to a flat architecture. This modular design allows each component to focus on its specific role, with the CPG Tuner modulating the base locomotion patterns and the Reflex Controller providing corrective adjustments based on the current state and environment.

In our work on facial expression transfer, we proposed a deep learning framework that uses the Facial Action Coding System (FACS) as an intermediate representation of facial expressions. This allowed us to map Action Units (AUs) detected from input images or videos to corresponding muscle activations in the biomechanical model. We also developed a data generation pipeline that synthesizes a large dataset of facial expressions and uses OpenFace to extract the corresponding AUs for labeling. This dataset was used to train our expression transfer network, allowing it to learn the mapping between AUs and muscle activations.

We then presented a real-time expression transfer pipeline that combines our trained network with OpenFace for AU detection and the biomechanical model for expression synthesis. This pipeline can transfer facial expressions and head poses from arbitrary input images or videos to our biomechanical model, producing anatomically consistent animations. We performed experiments on the KDEF, CK, and RAVDESS datasets to evaluate the effectiveness of our approach. The results showed that our method can effectively transfer a range of facial expressions and head poses.

6.2 Limitations and Future Work

While our approach achieved promising results in the neuromuscular control of biomechanical human musculoskeletal models, it has several limitations that open up potential avenues for future research.

6.2.1 Central Pattern Generator for Bipedal Locomotion Control

One limitation of our current approach is that it focuses primarily on bipedal locomotion, specifically walking. While this is a fundamental locomotion skill, human beings are capable of a much wider range of repetitive motions, such as climbing, swimming, and dancing. Additionally, our work does not address upper-body movements, such as arm swinging, in bipedal locomotion, which play a crucial role in maintaining balance. Extending our framework to encompass these additional locomotion modes and incorporating helpful upper-body motions can greatly enhance the completeness and versatility of our method.

Another limitation is that our experiments primarily focus on locomotion on flat, even surfaces. Real-world environments often present uneven terrains, obstacles, and other challenges that require more advanced adaptation and control strategies. A promising direction for future research is to test our approach in adapting to uneven terrains or external perturbations.

Furthermore, integrating higher-level planning and decision-making capabilities into our locomotion control framework is an important area for future research. Currently, our model relies on externally provided goals and cannot autonomously adjust to complex environments. A notable related work is (Peng et al., 2022), where the authors achieve high-level motion planning via hierarchical reinforcement learning.

6.2.2 Learning-Based Facial Expression Transfer From Images and Videos

Although our approach achieves promising results, several limitations remain in our work to date. First, while the pioneering framework and musculoskeletal model developed by Lee and Terzopoulos (2006) provided a solid foundation for our work, its limited anatomical fidelity and primitive facial soft tissue simulation techniques hinder the achievement of higher levels of realism. In our experiments, we observed that the model occasionally generates wrinkles or artifacts that do not accurately represent the reference expressions, which negatively impacts visual quality. Enhancing our biomechanical face model is a potential focus in the future, and the *Phace* framework (Ichim et al., 2017) is a noticeable related work that applies advanced physics-based simulation methods to improve biomechanical accuracy.

In addition, our method focuses on transferring facial expressions and head poses but does not consider other crucial aspects of facial animation, such as eye gaze and lip synchronization. These elements play a crucial role in creating fully expressive and lifelike facial animations. Incorporating these elements would require additional modeling and control techniques in our biomechanical face model, as well as the integration of relevant data sources, such as eye tracking and audio signals.

Moreover, our current approach relies on OpenFace for AU detection, which can be regarded as a limitation. Although OpenFace provides state-of-the-art performance in AU recognition, it struggles with certain challenging expressions or lighting conditions. An interesting direction for future work would be to explore end-to-end learning approaches that directly map input images or videos to muscle activations. Comparing the performance of such approaches with our method, which leverages an expert knowledge-based intermediate representation, would be valuable in assessing the trade-offs between the two paradigms.

APPENDIX A

Facial Expression Datasets

In our experiments, we utilized four widely-used facial expression datasets: the Karolinska Directed Emotional Faces (KDEF) dataset, the Cohn-Kanade (CK) dataset, the Extended Cohn-Kanade (CK+) dataset, and the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) dataset. These datasets contain a variety of facial expressions from multiple subjects and provide a comprehensive platform for evaluating our facial expression transfer method.

This appendix, as with the main chapter on facial expression transfer, is heavily borrowed from our publication (Zeng et al., 2021b).

A.1 Karolinska Directed Emotional Faces (KDEF)

The Karolinska Directed Emotional Faces (KDEF) dataset (Calvo and Lundqvist, 2008) consists of 4,900 pictures of human facial expressions. It features 70 subjects (35 female and 35 male) enacting all six basic facial expressions, namely neutral, joy, sadness, anger, fear, disgust, and surprise. Each expression performed by a subject is imaged from multiple directions. In our experiments, we used the KDEF dataset to evaluate the performance of our facial expression transfer method on a wide range of expressions and subjects.

A.2 Cohn-Kanade (CK) and Extended Cohn Kanade (CK+)

The Cohn-Kanade (CK) dataset (Kanade et al., 2000) is a widely-used benchmark dataset for facial expression recognition and analysis. It contains 486 sequences from 97 subjects, each sequence starting with a neutral expression and ending with a peak expression. The peak expressions are FACS coded, providing a ground truth for action unit activation.

The Extended Cohn-Kanade (CK+) dataset (Lucey et al., 2010) is an extension of the original CK dataset, containing 593 sequences from 123 subjects. Similar to the CK dataset, each sequence starts with a neutral expression and ends with a peak expression. The main difference is that the CK+ dataset includes more sequences and subjects, as well as additional metadata such as emotion labels and action unit intensity scores.

A.3 Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) dataset (Livingstone and Russo, 2018) is a multimodal database containing speech and song recordings from 24 professional actors (12 female and 12 male). Each actor performs 60 speech trials and about 44 song trials with 8 emotions (neutral, calm, happy, sad, angry, fear, disgust, and surprise) and 2 intensities (normal and strong). This yields a total of 2,452 video files for the complete dataset. In our experiments, we used the RAVDESS dataset to evaluate the performance of our facial expression transfer method on dynamic expressions in video sequences.

REFERENCES

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*. 35

Baltrušaitis, T., Zadeh, A., Lim, Y. C., and Morency, L.-P. (2018). OpenFace 2.0: Facial behavior analysis toolkit. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 59–66. 55, 56

Calvo, M. G. and Lundqvist, D. (2008). Facial expressions of emotion (KDEF): Identification under different display-duration conditions. *Behavior Research Methods*, 40(1):109–115. 61, 71

Cohn, J. F., Ambadar, Z., and Ekman, P. (2007). Observer-based measurement of facial expression with the facial action coding system. *The Handbook of Emotion Elicitation and Assessment*, 1(3):203–221. 4

DiLorenzo, P. C. (2009). *Breathing, laughing, sneezing, coughing: Model and control of an anatomically inspired, physically-based human torso simulation*. PhD thesis, University of California, Riverside. 7

DiLorenzo, P. C., Zordan, V. B., and Sanders, B. L. (2008). Laughing out loud: Control for modeling anatomically inspired laughter using audio. In *ACM SIGGRAPH Asia 2008 Papers*, pages 1–8. 7

Dzeladini, F., Ait-Bouziad, N., and Ijspeert, A. (2018). CPG-based control of humanoid robot locomotion. In *Humanoid Robotics: A Reference*, pages 1–35. Springer. 10

Edwards, P., Landreth, C., Fiume, E., and Singh, K. (2016). JALI: An animator-centric viseme model for expressive lip synchronization. *ACM Transactions on Graphics (TOG)*, 35(4):1–11. 11

Ekman, P. and Friesen, W. V. (1978). Facial action coding system. *Environmental Psychology and Nonverbal Behavior*. 4, 11, 54

Featherstone, R. (2014). *Rigid Body Dynamics Algorithms*. Springer. 8

Gams, A., Ijspeert, A. J., Schaal, S., and Lenarčič, J. (2009). On-line learning and modulation of periodic movements with nonlinear dynamical systems. *Autonomous Robots*, 27(1):3–23. 31, 32

Gonzalez, R. V., Buchanan, T. S., and Delp, S. L. (1997). How muscle architecture and moment arms affect wrist flexion-extension moments. *Journal of Biomechanics*, 30(7):705–712. 16

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft Actor-Critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR. 39

Hatamizadeh, A., Hoogi, A., Sengupta, D., Lu, W., Wilcox, B., Rubin, D., and Terzopoulos, D. (2019). Deep active lesion segmentation. In *Machine Learning in Medical Imaging: 10th International Workshop, MLMI 2019, Held in Conjunction with MICCAI 2019, Shenzhen, China, October 13, 2019, Proceedings 10*, pages 98–105. Springer. xiii

Hill, A. V. (1938). The heat of shortening and the dynamic constants of muscle. *Proceedings of the Royal Society of London. Series B-Biological Sciences*, 126(843):136–195. 17

Holzinger, A., Langs, G., Denk, H., Zatloukal, K., and Müller, H. (2019). Causability and explainability of artificial intelligence in medicine. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(4):e1312. 9

Ichim, A.-E., Kadleček, P., Kavan, L., and Pauly, M. (2017). Phace: Physics-based face modeling and animation. *ACM Transactions on Graphics (TOG)*, 36(4):1–14. 69

Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642–653. 9

Ijspeert, A. J., Crespi, A., Ryczko, D., and Cabelguen, J.-M. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420. 5, 9

Jiang, Y., Van Wouwe, T., De Groote, F., and Liu, C. K. (2019). Synthesis of biologically realistic human motion using joint torque actuation. *ACM Transactions on Graphics (TOG)*, 38(4):1–12. 9

Kähler, K., Haber, J., Yamauchi, H., and Seidel, H.-P. (2002). Head shop: Generating animated head models with anatomical structure. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 55–63. 7

Kanade, T., Cohn, J. F., and Tian, Y. (2000). Comprehensive database for facial expression analysis. In *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 46–53. IEEE. 61, 72

Kidziński, Ł., Mohanty, S. P., Ong, C., Hicks, J., Francis, S., Levine, S., Salathé, M., and Delp, S. (2018). Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning. In Escalera, S. and Weimer, M., editors, *NIPS 2017 Competition Book*. Springer, Springer. 25, 43, 45

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 58

Kitagawa, M. and Windsor, B. (2020). *MoCap for artists: Workflow and techniques for motion capture*. CRC Press. 2

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444. 9

Lee, S., Park, M., Lee, K., and Lee, J. (2019). Scalable muscle-actuated human simulation and control. *ACM Transactions on Graphics (TOG)*, 38(4):1–13. 7, 8, 9, 11, 46, 47

Lee, S.-H., Sifakis, E., and Terzopoulos, D. (2009). Comprehensive biomechanical modeling and simulation of the upper body. *ACM Transactions on Graphics (TOG)*, 28(4):1–17. 4, 8, 14, 16, 19

Lee, S.-H. and Terzopoulos, D. (2006). Heads up! biomechanical modeling and neuromuscular control of the neck. In *ACM SIGGRAPH 2006 Papers*, pages 1188–1198. 5, 7, 8, 10, 14, 18, 20, 24, 57, 64, 69

Lee, Y., Terzopoulos, D., and Waters, K. (1995). Realistic modeling for facial animation. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 55–62. 7, 21

Li, H., Yu, J., Ye, Y., and Bregler, C. (2013). Realtime facial animation with on-the-fly correctives. *ACM Transactions on Graphics (TOG)*, 32(4):42–1. 11

Livingstone, S. R. and Russo, F. A. (2018). The Ryerson audio-visual database of emotional speech and song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. *PloS One*, 13(5):e0196391. 72

Ljung, L. and Söderström, T. (1983). *Theory and Practice of Recursive Identification*. MIT press. 33

Lu, W. and Liu, L. (2020). Surface reconstruction via cooperative evolutions. *Computer Aided Geometric Design*, 77:101831. xiii

Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., and Matthews, I. (2010). The extended Cohn-Kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition—Workshops*, pages 94–101. IEEE. 61, 72

Merel, J., Botvinick, M., and Wayne, G. (2019). Hierarchical motor control in mammals and machines. *Nature Communications*, 10(1):1–12. 11

Mori, M., MacDorman, K. F., and Kageki, N. (2012). The uncanny valley [from the field]. *IEEE Robotics & Automation Magazine*, 19(2):98–100. 1

Nakada, M., Zhou, T., Chen, H., Weiss, T., and Terzopoulos, D. (2018). Deep learning of biomimetic sensorimotor control for biomechanical human animation. *ACM Transactions on Graphics (TOG)*, 37(4):1–15. 4, 7, 9, 24

Ng-Thow-Hing, V. (2001). *Anatomically-based models for physical and geometric reconstruction of humans and other animals*. PhD thesis, Department of Computer Science, University of Toronto. 17

Nordin, M. (2020). *Basic Biomechanics of the Musculoskeletal System*. Lippincott Williams & Wilkins. 7

NVIDIA Corporation (2023). NVIDIA Isaac Sim User Manual. `https://docs.omniverse.nvidia.com/isaacsim/latest/index.html`. 27

Parke, F. I. (1972). Computer generated animation of faces. In *Proceedings of the ACM Annual Conference—Volume 1*, pages 451–457. 10

Parr, L. A., Waller, B. M., Vick, S. J., and Bard, K. A. (2007). Classifying chimpanzee facial expressions using muscle action. *Emotion*, 7(1):172. 55

Peng, X. B., Abbeel, P., Levine, S., and Van de Panne, M. (2018). DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):1–14. 9, 11, 46, 47, 49

Peng, X. B., Berseth, G., and Van de Panne, M. (2016). Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 35(4):1–12. 11

Peng, X. B., Berseth, G., Yin, K., and Van De Panne, M. (2017). DeepLoco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4):1–13. 11

Peng, X. B., Guo, Y., Halper, L., Levine, S., and Fidler, S. (2022). ASE: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions on Graphics (TOG)*, 41(4):1–17. 69

Peng, X. B. and Van De Panne, M. (2017). Learning locomotion skills using DeepRL: Does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–13. 9

Phinyomark, A., Petri, G., Ibáñez-Marcelo, E., Osis, S. T., and Ferber, R. (2018). Analysis of big data in gait biomechanics: Current trends and future directions. *Journal of Medical and Biological Engineering*, 38:244–260. 8

Platt, S. M. and Badler, N. I. (1981). Animating facial expressions. In *Proceedings of the 8th Annual Conference on Computer Graphics and Interactive Techniques*, pages 245–252. 55

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8. 50

Rajagopal, A., Dembia, C. L., DeMers, M. S., Delp, D. D., Hicks, J. L., and Delp, S. L. (2016). Full-body musculoskeletal model for muscle-driven simulation of human gait. *IEEE Transactions on Biomedical Engineering*, 63(10):2068–2079. 7

Salisu, S., Ruhaiyem, N. I. R., Eisa, T. A. E., Nasser, M., Saeed, F., and Younis, H. A. (2023). Motion capture technologies for ergonomics: A systematic literature review. *Diagnostics*, 13(15):2593. 2

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015a). Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897. PMLR. 38

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2015b). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*. 38

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*. 11, 38

Seth, A., Hicks, J. L., Uchida, T. K., Habib, A., Dembia, C. L., Dunne, J. J., Ong, C. F., DeMers, M. S., Rajagopal, A., Millard, M., et al. (2018). OpenSim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. *PLoS Computational Biology*, 14(7):e1006223. 25

Si, W., Lee, S.-H., Sifakis, E., and Terzopoulos, D. (2014). Realistic biomechanical simulation and control of human swimming. *ACM Transactions on Graphics (TOG)*, 34(1):1–15. 7, 10, 24, 32, 49

Sifakis, E., Neverov, I., and Fedkiw, R. (2005). Automatic determination of facial muscle activations from sparse motion capture marker data. In *ACM SIGGRAPH 2005 Papers*, pages 417–425. 7, 10

Song, S., Kidziński, Ł., Peng, X. B., Ong, C., Hicks, J., Levine, S., Atkeson, C. G., and Delp, S. L. (2021). Deep reinforcement learning for modeling human locomotion control in neuromechanical simulation. *Journal of Neuroengineering and Rehabilitation*, 18:1–17. 7, 30

Sueda, S., Kaufman, A., and Pai, D. K. (2008). Musculotendon simulation for hand animation. In *ACM SIGGRAPH 2008 Papers*, pages 1–8. 7

Sumathi, C., Santhanam, T., and Mahadevi, M. (2012). Automatic facial expression analysis: A survey. *International Journal of Computer Science and Engineering Survey*, 3(6):47. 55

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press. 40

Sylvester, A. D., Lautzenheiser, S. G., and Kramer, P. A. (2021). A review of musculoskeletal modelling of human locomotion. *Interface Focus*, 11(5):20200060. 28

Taga, G. (1995). A model of the neuro-musculo-skeletal system for human locomotion: I. emergence of basic gait. *Biological Cybernetics*, 73(2):97–111. 10

Tjoa, E. and Guan, C. (2020). A survey on explainable artificial intelligence (XAI): Toward medical XAI. *IEEE Transactions on Neural Networks and Learning Systems*, 32(11):4793–4813. 9

Tsang, W., Singh, K., and Fiume, E. (2005). Helping hand: An anatomically accurate inverse dynamics solution for unconstrained hand motion. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 319–328. 7

Van Den Bogert, A. J., Blana, D., and Heinrich, D. (2011). Implicit methods for efficient musculoskeletal simulation and optimal control. *Procedia Iutam*, 2:297–316. 8

Van Nierop, O. A., van der Helm, A., Overbeeke, K. J., and Djajadiningrat, T. J. (2008). A natural human hand model. *The Visual Computer*, 24:31–44. 7

Vijayakumar, S. and Schaal, S. (2000). Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, volume 1, pages 288–293. Morgan Kaufmann. 32

Wade, L., Needham, L., McGuigan, P., and Bilzon, J. (2022). Applications and limitations of current markerless motion capture methods for clinical gait biomechanics. *PeerJ*, 10:e12995. 2

Waters, K. (1987). A muscle model for animation three-dimensional facial expression. *ACM SIGGRAPH Computer Graphics*, 21(4):17–24. 10

Weise, T., Bouaziz, S., Li, H., and Pauly, M. (2011). Realtime performance-based facial animation. *ACM Transactions on Graphics (TOG)*, 30(4):1–10. 11

Winters, J. M. (1990). Hill-based muscle models: A systems engineering perspective. In *Multiple Muscle Systems: Biomechanics and Movement Organization*, pages 69–93. Springer. 17

Wisdom, K. M., Delp, S. L., and Kuhl, E. (2015). Use it or lose it: Multiscale skeletal muscle adaptation to mechanical stimuli. *Biomechanics and Modeling in Mechanobiology*, 14:195–215. 18

Wojdeł, A. and Rothkrantz, L. J. (2005). Parametric generation of facial expressions based on FACS. *Computer Graphics Forum*, 24(4):743–757. 54, 55

Zajac, F. E. (1989). Muscle and tendon: Properties, models, scaling, and application to biomechanics and motor control. *Critical Reviews in Biomedical Engineering*, 17(4):359–411. 13

Zeng, X. S., Dwarakanath, S., Lu, W., Nakada, M., and Terzopoulos, D. (2021a). Facial expression transfer from video via deep learning. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–2. xiii, 14

Zeng, X. S., Dwarakanath, S., Lu, W., Nakada, M., and Terzopoulos, D. (2021b). Neuromuscular control of the face-head-neck biomechanical complex with learning-based expression transfer from images and videos. In *Advances in Visual Computing: 16th International Symposium, ISVC 2021, Virtual Event, October 4-6, 2021, Proceedings, Part I*, pages 116–127. Springer. xiii, 14, 54, 59, 71

Zhou, T. (2019). *Core Training: Learning Deep Neuromuscular Control of the Torso for Anthropomimetic Animation.* PhD thesis, Computer Science Department, University of California, Los Angeles. 7, 9, 24, 25