Capturing and Simulating Realistic Pedestrian Movements

Janis Sprenger

Dissertation zur Erlangung des Grades des
Doktors der Ingenieurwissenschaften (Dr.-Ing.)
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

Saarbrücken
2025

Tag des Kolloquiums	02.06.2025
Dekan	Prof. Dr. Roland Speicher
Vorsitzender	Prof. Dr.-Ing. Thorsten Herfet
Berichterstatter	Prof. Dr.-Ing. Philipp Slussalek
	Prof. Dr. Verena Wolf
Akademischer Mitarbeiter	Dr.-Ing. Christian Müller

Abstract

The motions of pedestrians are complex and highly variable. Autonomous vehicles must interact safely and reliably with pedestrians, but real-world testing poses ethical challenges. Thus, a digital simulation is required. While sensor simulation has progressed significantly, pedestrian motion simulation still relies on unrealistic game animations that do not visualize the necessary cues for identifying crossing decisions correctly.

This thesis introduces new methods to capture pedestrian motion required to simulate realistic pre-crash scenarios. Several studies with over 150 participants in real and virtual environments analyzed the motion and behavior before a crossing. The studies include comparing cultural differences between German and Japanese pedestrians and investigating experiments with children.

Several data-driven motion synthesis approaches are presented: extending phase-functioned neural networks to recreate natural styles of male and female pedestrians, improving mixture-of-expert models to increase the natural variance in locomotion, and extending them with multi-action control for gaze, foot placement, and hand movement. A generative model is presented that uses reinforcement learning to control and animate pedestrians avoiding each other in crowds.

A solution to retarget the generated motions to different character meshes is presented. The MOSIM Framework is introduced to integrate pedestrians into the famous driving simulator CARLA.

Kurzfassung

Fußgängerbewegungen sind komplex und variabel. Autonome Fahrzeuge müssen sicher und zuverlässig mit Fußgängern interagieren, aber Tests mit realen Menschen sind ethisch schwierig, weshalb eine digitale Simulation benötigt wird. Während die Sensor-Simulation bereits fortgeschritten ist, basieren Fußgängeranimationen meist auf unrealistischen Spiel-Animationen ohne die notwendigen Hinweisreize um eine Überquerungsentscheidung korrekt zu identifizieren.

Diese Dissertation stellt Methoden zur Erfassung und Simulation von Fußgängerbewegungen in Pre-Crash-Situationen vor. In mehreren Studien mit über 150 Teilnehmern in realen und virtuellen Umgebungen wurden die Bewegungen und das Verhalten analysiert, unter anderem zwischen deutschen und japanischen Fußgängern und von Kindern.

Mehrere datengetriebene Animationsmodelle werden vorgestellt: ein phasen-interpolierendes neuronales Netz wurde erweitert, um natürliche Bewegungen von Männern und Frauen zu erzeugen. Mixture-of-Expert Modelle wurden erweitert, um die Varianz der erzeugten Bewegung zu erhöhen und die Kontrolle der Blickrichtung, der Fußplatzierung und von Handbewegungen zu ermöglichen. Mithilfe von verstärkendem Lernen wird ein generatives Modell kontrolliert, um Ausweichbewegungen von Fußgängern in Menschenmengen zu simulieren.

Ein Ansatz zum Übertragen von Bewegungen auf verschiedene Charaktermodelle wurde implementiert. Das MOSIM Framework wird präsentiert und dazu genutzt, Fußgänger in dem CARLA-Fahrsimulator zu animieren.

Acknowledgements

I am incredibly thankful to Philipp for the opportunity to work on this topic, his support in my endeavors, and his guidance. However, I could not have taken this journey without Klaus, Matthias, and Christian. Each taught me valuable lessons that enabled me to write my papers and this dissertation.

I am very grateful to all of the members of my team. Klaus, Han, Erik, Somayeh, Noshaba, Anindita, Rui, and Jan always had an open ear to discuss ideas, provided valuable feedback, and helped with proofreading. I was very happy when Lorena joined, and I am grateful that she managed to bring our experiments to a much higher standard. I am also thankful to the other teams of ASR, in particular to André, with whom I collaborated intensively over the past years, to Nils, who was always available when I needed him, and to Igor, who was looking for ways to bring our tech to the market. Special thanks go to all of my students and HiWis for their excellent work and support, particularly Ulan, Niklas, Rolando, Chirag, Helena, Sina, Annika, Vannessa, and Mina. Special thanks go to Léa for always supporting me at DFKI.

I am deeply indebted to the HARC team at AIST for the opportunity to join their team for two months and for their phenomenal hospitality, particularly Yoshi, Saori, Shoma, Gaku, and Kengo. あなたと過ごした時間や私たちの経験に心から感謝しています。それを決して忘れることはなく、いつも感謝の気持ちを抱いています。

I want to express my deepest thanks to my family for their unconditional support and my dad for proofreading the dissertation in its entirety. Thanks also to all of my friends for their support and that you endured my mood swings while writing the manuscript. And last, to my wife Katrin. You are the love of my life and without your support, your feedback, and your motivation, I would not have survived this dissertation.

List of Abbreviations

3D	Three-dimensional
AAA	American Automotive Association
ADAPT	Agent development and prototyping testbed
ADAS	Advanced driver assistant systems
AI	Artificial intelligence
AIST	(National Institute of) Advanced Industrial Science and Technology
AIToC	Proper name of the research project "AIToC - Artificial Intelligence supported Tool Chain in Manufacturing Engineering"
AJAN	Accessible Java Agent Nucleus
et al.	Et Alii (Latin); and others
AMD	Advanced Micro Devices Inc.
ANOVA	Analysis of variance
ASM	Archetypal style model
BMBF	Bundesministerium für Bildung und Forschung
BMI	Body mass index
BML	Behavior Markup Language
BMWK	Bundesministerium für Wirtschaft und Klimaschutz
BVH	Biovision hierarchy
CAD	Computer-aided design
cf.	Confer (Latin); compare to, compare with
COVID-19	Coronavirus disease 2019
CPU	Central processing unit
cVAE	Conditional variational autoencoder
DFKI	Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
DLL	Dynamic link library
e.g.	Exempli gratia (Latin); for the sake of an example
ELU	Exponential linear unit
ERD	Encoder-recurrent-decoder
etc.	Et cetera (Latin); and the rest
EU	European Union
FABRIK	Forward And Backward Reaching Inverse Kinematics
fbx	Autodesk Filmbox, a proprietary fileformat
ff.	And the following pages
FK	Forward kinematics
FMI	Functional Mockup Interface
FMU	Functional Mockup Unit
FOV	Field of view

fps	Frames per second
GAN	Generative adversarial network
GKM	General knowledge model
GmbH	Gesellschaft mit beschränkter Haftung
GMM	Gaussian mixture model
GP	Gaussian process
GPS	Global Positioning System
GPT	Generative pretrained transformer
GPU	Graphical processing unit
gRPC	gRPC Remote Procedure Calls
GRU	Gated recurrent unit
HIL	Hardware in the loop
HMD	Head-mounted display
HMM	Hidden Markov model
HTTP	Hypertext transfer protocol
i.e.	Id est (Latin); that is
IK	Inverse kinematics
IMU	Inertia measurement unit
ITEA	Information Technology for European Advancement
KL	Kullback-Leibler
LIDAR	Light imaging, detection and ranging
LMA	Laban movement analysis
LSM	Linear style model
LSTM	Long short-term memory
MANN	Mode adaptive neural network
MDP	Markov decision process
MIL	Model in the loop
ML	Machine learning
MMU	Modular model units
MOSIM	Proper name (based on the project name "MOSIM - End-to-end Digital Integration on MOdular SIMulation of Natural Human Motions")
MQTT	Message queue (MQ) telemetry transport
MSE	Mean squared error
MTM	Methods-Time Measurement
NCAP	New Car Assessment Program
No.	Number
NPC	Non-player character
NSM	Neural state machine
ONNX	Open neural network exchange
OPC-UA	Open Platform Communications - Unified Architecture

OpenGL	Open graphics library
PC	Personal computer
PCA	Principle component analysis
PFG	Pedestrian flashing green
PFNN	Phase-functioned neural network
PPO	Principle policy optimization
protobuf	Protocol buffers
R2I	Retarget to intermediate
R2T	Retarget to target
RAM	Random access memory
RBM	Restricted Boltzmann machine
RDF	Resource description format
RL	Reinforcement learning
RNN	Recurrent neural network
ROS	Robot Operating System
RPC	Remote procedure call
SAE	Society of Automotive Engineers
SIL	Software in the loop
SLAM	Simultaneous localization and mapping
SMPL	Skinned multi-person linear model
SPARQL	SPARQL Protocol and RDF Query Language
StVO	Straßenverkehrsordnung
TCP	Transmission control protocol
TTC	Time to collision
USA	United States of America
VAE	Variational autoencoder
VIL	Vehicle in the loop
VINN	Variational interpolating neural networks
VQ	Vector quantized
VR	Virtual reality
VRAM	Video random access memory
XML	Extensible markup language
XR	Extended reality
ZF	ZF Friedrichshafen AG
ZIM	Zentrales Innovationsprogramm Mittelstand

Content

Chapter 1
Introduction

Walking is the most popular mode of transportation in Europe after driving [Com+22]. Pedestrians move with high variability, not strictly adhering to traffic regulations or "traffic lanes". They are the weakest participants in urban traffic (33% of all urban fatalities in 2022 [Com24]) and thus require the most attention from other participants, like vehicles. Modern vehicles are equipped with many advanced driver-assistant systems (ADAS). These systems aim to reduce potential harm to both, the vehicle occupants and other traffic participants, including pedestrians. Since 2022, highly automated breaking assistants have been required by law in all new vehicle types in Europe, and all newly registered vehicles in 2024 [PC22]. More assistant systems and extensive testing and validation with pedestrian scenarios will be required starting from 2024 (new types) and 2026 (new registrations), respectively [PC22]. Fully autonomous vehicles are already starting to drive on German highways but are not yet approved for widespread use in urban environments [Bun23].

Several aspects complicate developing and testing of autonomous vehicle features for pedestrian safety in in-vivo environments. Real driving data, including vehicle and sensor data, is expensive to capture and annotate. Privacy rights can complicate the procedure, especially with unknown pedestrians outside of the vehicle. These rights are especially important in recordings of accidents and near misses. Fortunately, these events rarely occur, but unfortunately, they are crucial for deriving testing scenarios and training data-driven systems. Testing a novel vehicle feature with real pedestrians is unethical, as the safety of such pedestrians cannot be ensured. Traditionally, testing is performed with humanoid puppets, dragged along a predefined path with a robotic system on testing grounds, with newer systems moving arms and legs uncoordinated. However, the reduced complexity of the testing grounds and the limited movement of the kinematic puppet does not reflect real environments and makes the usage of real cues indicating pedestrian intention and behavior impossible.

To tackle the lack of real data and the infeasibility of real testing, it has been established over the past years that the utilization of digital simulations for both, training and evaluation, will be required to allow the development and certification of new ADAS systems [Dah+19]. The simulation of virtual vehicles is actively utilized in production. There are advanced approaches to allow for realistic simulations of vehicles and vehicle components in digital twins [Qui+09; EM11; Mat21], up to real hardware in a virtual testing simulation (hardware in the loop, HIL) [MTH22]. However, the behavior and motions of pedestrians are highly complex and not yet adequately

Figure 1.1: Visualization of a pre-crash scenario inside a virtual replica of the Eisenbahnstraße in Saarbrücken.

represented in production software. In addition to the variability in their outer appearance, their crossing behavior and the motions executed to perform the crossing differ between individuals. Real testing is only feasible in limited scenarios, ensuring the safety of real people. Thus, a realistic simulation for virtual testing is required. **In this dissertation, the simulation of pedestrian agents for evaluating ADAS systems is the primary application goal**. Unlike evaluations of passive safety systems, the simulation is focused on the time frame before an actual crash occurs (pre-crash simulations). An example of a pre-crash scenario rendered inside a digital twin of an actual road is shown in Figure 1.1.

Pedestrian simulations can be categorized into three different levels of abstraction: (i) microscopic, (ii) macroscopic, and (iii) mesoscopic simulations.

In **microscopic simulations**, highly accurate digital human models are used to analyze the physical interaction between cars and pedestrians. These are primarily used for in-crash simulations, for example, to investigate the risk of fatal injuries based on the design of the car chassis. They are ideal for the simulation of the actual collision of the vehicle with the pedestrian, e.g., to test passive safety features. Still, they are not suited for pre-crash simulations as they do not focus on an accurate reproduction of anticipatory movements indicating a crossing decision.

Off-road activities and large-scale simulations are of interest in **macroscopic simulations**. Off-road activities contain the topics of where a journey starts and where it ends or which mode of transportation is utilized. The scale is usually very large, focusing, for example, on crowd movements and interaction of crowds with the traffic and with themselves. These simulations are of primary interest to city planners, the generation of evacuation plans, or event managers. The level of detail is too coarse for pre-crash simulations, although highly relevant for smart and integrated cities and global route choices of autonomous vehicles.

In **mesoscopic simulations**, pedestrians' road-related behavior and movements are of high relevance. This includes pedestrians' behavior, actions, and movements on the sidewalk and the roadway. The reason why and where the journey is planned is only relevant to the current state of mind (e.g., urgency, attentiveness, drunkenness, etc.). The focus is primarily on the actual crossing situations. The simulations must be realistic and accurate enough for the sensors in the car but not necessarily on a detailed biomechanical level. Ideally, the sensor-specific representation of the behavior and the motion should be indistinguishable from the data received in real environments. A mesoscopic perspective on pedestrian simulation needs to be considered for the training and validation of most ADAS systems. Therefore, **this dissertation focuses solely on the mesoscopic simulation level**.

For pre-crash simulations, a controllable simulation of configurable human models that are realistic, variational, and explainable is required. **Uncontrolled simulations** might result in emerging behavior and realistic critical scenarios. However, the simulation might need to run indefinitely long to create a specific critical scenario. Thus, uncontrolled simulation does not bring any major benefit compared to real data. Similar to coverage criteria in software testing, scenario coverage should be targeted, and failure cases should be systematically investigated. As such, the simulation models have to be **configurable** to simulate behavior and motions for pedestrians of different ages, genders, sizes, cultures, intentions, or behavioral traits. A sufficient level of **realism** is required, including realistic trajectories, spontaneous actions, and body movements. This includes decision errors and shortcomings of pedestrians that naturally occur, such as the misestimation of vehicle speed and, thus, the decision to cross a critical gap in traffic. The simulation should sufficiently represent the natural **variation** in movements to support realism. This includes the inter-variation between pedestrians as well as the intra-variation of an individual. The simulation should generate **explainable behavior**, most of all for uncommon and critical scenarios. For example, "suicidal pedestrian agents" [Yan+23] that are configured to target a collision with the vehicle at all costs by suddenly running in front of the approaching vehicle can be simulated. However, suicidal behavior occurs rarely, and thus, only simulating this type of behavior would introduce a heavy bias to the ADAS model. Hence, the generated behavior should be motivated, e.g., by considering and simulating why the sudden change in trajectory and velocity was performed; including, but not limited to, suicidal behavior.

1.1 Conceptual Architecture of Pedestrian Simulation

Generating realistic simulations of pedestrian agents requires more than a trajectory generator or an animation model. Based on the agent-simulation paradigm [Wei99], the following conceptual architecture for pedestrian agents in pre-crash simulations is proposed. The modules **Sense**, **Think**, and **Act** are required for the core simulation of a pedestrian agent (compare as well [RN16, p. 34 ff.]). It is required to **Understand** the reasons and the intricacies of pedestrian behavior and motion for a correct configuration, training, and evaluation of simulation models. The module

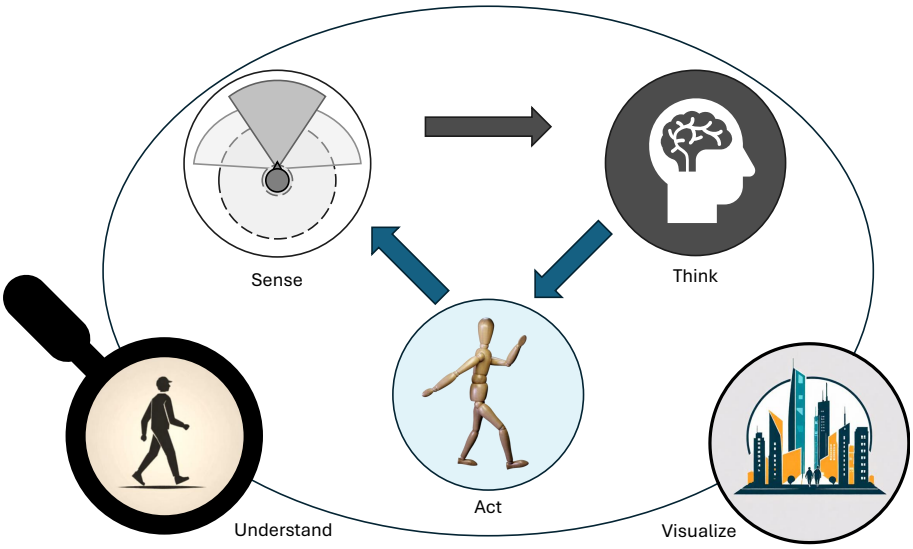


Figure 1.2: Visualization of different modules of the conceptual architecture for pedestrian simulation. This work focuses on the **Understand** (Chapter 3), **Act** (Chapter 4), and **Visualize** (Chapter 5). While **Sense** and **Think** are important aspects of an agent, they are outside of the scope of this dissertation.[1]

Visualize comprises a 3D engine that is required to generate critical scenario simulations and to create the sensor input for autonomous vehicles. The agents need to be enabled to utilize the visualization engine. An overview of this conceptual architecture is depicted in Figure 1.2. This dissertation focuses on the modules **Understand**, **Act**, and **Visualize**, in particular for articulated agent motions. The work is integrated into a greater effort for pedestrian simulation in the research department Agents and Simulated Reality at the German Research Center for Artificial Intelligence (DFKI) and thus, research in **Sense** and **Think** modules of pedestrian agents are conducted by others and have been connected to this work on several occasions. Subsequently, the different modules are described in detail.

1.1.1 Understand: Pedestrians Behavior and Motion

It is important to have a solid understanding of pedestrian behavior and motion, including the necessary training data, in order to simulate pedestrians realistically. Traffic psychology research provides different information on varying granularity, with most information on decision-making, like which gap between vehicles would still be considered safe to cross the road [Oxl+97; Con+98; Oxl+05; LC07; Sch+12]. On this behavioral level, there are reports of cultural, age, and gender differences that need to be considered [Hel+21]. The information provided by traffic psychology is usually not in a format that supports model training but can be utilized to configure cognitive models. In addition, the same methods for observing and evaluating real pedestrians can be employed to evaluate the realism of the generated simulation. However, limited knowledge is

[1]The images depicting analysis and visualization are generated with hotpot.ai (`https://hotpot.ai/logo-generator`). The image depicting the articulated motion is a render of the wooden mannequin 3D Asset by James Wright (`https://skfb.ly/oyxpC`).

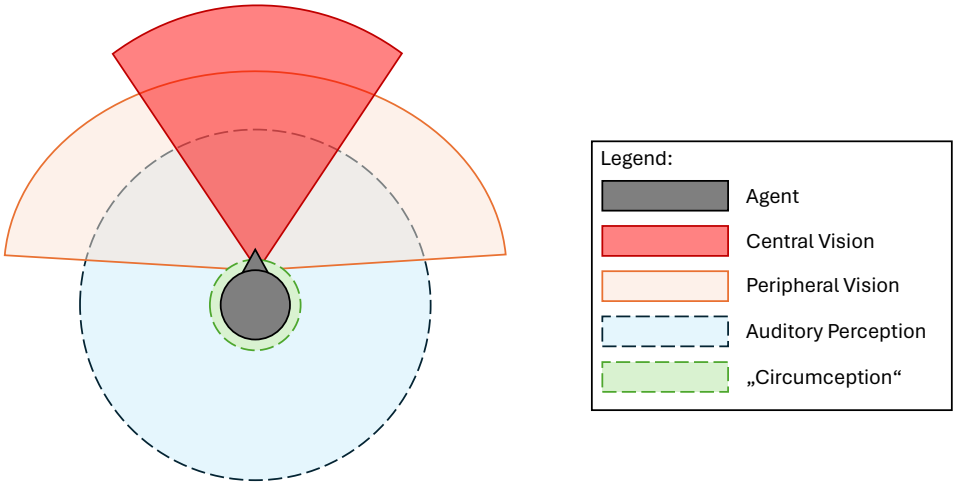


Figure 1.3: Different perception types for a virtual agent. "Circumception" gathers different aspects of far peripheral vision, tactile perception, proprioception, and short-term memory.

available about pedestrians' actual body movements. Context-specific full-body motion data is missing almost completely and is barely analyzed [KRO18].

One of the core topics of this dissertation is to increase the knowledge of pedestrian behavior and to generate the necessary data to train pedestrian simulation models. The respective work on this topic can be found in Chapter 3.

1.1.2 Sense: Simulation of Perception

Throughout the simulation, the digital pedestrian needs to "perceive" the environment. Unlike the real world, a digital environment provides more information than raw images and sound, including state information (e.g., speed of the approaching car), semantic information (e.g., the object is a car), and context information (e.g., this is a one-way street). Learning from first principles is neither required nor beneficial when considering the background information available. Hence, the senses do not only include raw visual and auditory information but also semantics, trajectory information of others (e.g., where they came from, where they are going), and ego state information (e.g., the ego-pedestrian standing on the street or the sidewalk). In order to execute natural behavior, the perception of the immediate environment ("circumception") is required, combining auditory, (peripheral) visual, and tactile perception and short-term memory processes. Although almost completely irrelevant to behavioral planning, it is highly important to incorporate this perception into the motion synthesis. For instance, the perception of ground height is important when placing the feet. However, constantly watching the curb is not required, as humans construct a mental model of their immediate surroundings to perform the correct placement. The different perception types and their ranges are visualized in Figure 1.3.

Concepts of circumception will be partially utilized in Chapter 4 as part of the neural network input for motion simulation, but other perceptions will not be investigated in detail in this dissertation.

1.1.3 Think: Behavior Planning and Cognition

Pedestrian behavior can be defined as the higher-level cognitive functions of planning, decision-making, prediction, anticipation, and attention modeling. Different methods exist to simulate these cognitive functions, from rule-based systems and behavior trees [Ant+21] to generative neural networks [Rem+23]. Behavior planning and decision-making usually operate with state-based and semantic information, like the past and current position and velocity of others in relation to their own state and goal. For a more realistic simulation of anticipation, prediction, and attention, the state-based information can be extended with perceptual information (e.g., video data) [Voz+23].

The connection of behavior planning and cognition models to motion simulation will be presented in Chapter 5. However, the implementation of new behavior planning and cognition models is not in the scope of this dissertation.

1.1.4 Act: Articulated Pedestrian Motions

The simulated behavior needs to be acted out by a virtual representation of a pedestrian to be perceived by the vehicle. This includes the requirement for full-body articulated motions [ZB23], the movement of the torso, legs, arms, and head. The behavior needs to be "acted" by the digital pedestrian in a realistic way to let an autonomous driving model perform a realistic prediction (e.g., pedestrian behavior prediction, path prediction, lateral and longitudinal control, etc.). In particular, this includes highly relevant cues to form a specific decision, like the walking and gaze direction, realistic rotations of the torso [Spr+19a], and explicit gestures [Mus+21]. Besides behavior visualization, motion can express additional information on age, physical state (e.g., drunkenness), or disabilities [Tro13]. Pedestrians with movement disorders, like Parkinson's disease, should be accurately detected. In case of motion freezing [Gil+92], the autonomous vehicle should not interpret the stationary pedestrian on the street as a misprediction but break and call for assistance. Human motion comprises of many variations and outliers. Simulating these variations is crucial for verifying whether ADAS systems are robust enough for real-world use cases. In extreme cases, a child performing a cartwheel next to the street should still be detected properly.

The simulation of articulated pedestrian motions is the core focus of this dissertation. The contributions to capturing motions will be discussed in Chapter 3, to the simulation of motion in Chapter 4, and to the display of motions in Chapter 5.

1.1.5 Visualize: Simulation Environments and 3D Engines

The pedestrians need to be simulated in a suitable simulation environment, which can also be utilized by the ADAS components. Multiple environments in different simulation engines are available, focusing on aspects like photo-realism, hardware support (e.g., HIL), or usability (e.g., [Dos+17]). As these systems are still new and the field is volatile, separating the actual pedestrian simulation from the simulation environment is beneficial. However, the requirements

Figure 1.4: A diverse set of pedestrian characters rendered in the CARLA driving simulator [Dos+17], showcasing variations in gender, height, age, weight, ethnicity, and clothing. Ensuring diversity in virtual environments is crucial for reducing bias in the final simulation.

are the same in any simulation environment. Digital pedestrians should appear realistic to the vehicle's sensor. For a mesoscopic simulation, it is essential to include a high variation in clothing, fashion styles, accessories, etc. Ethical fairness is highly important, as a balance in body size, skin tone, gender, age, and minority groups must be found to prevent overfitting or testing bias [Fab+22]. Moreover, deviation from the real-life distributions is required for generating corner cases, and the simulations need to be adjustable, for example, adding face masks to a simulation to evaluate the performance in a worldwide pandemic. Different types of pedestrian models are shown in Figure 1.4.

Unlike the entertainment industry, a very high level of detail – like subsurface scattering under the skin, simulation of pupil dilation, or the simulation of blood vessels [Got+18] – is unnecessarily complex due to the physical distance between vehicle and pedestrian and the resolution of typical sensors installed in autonomous vehicles.

The integration of motion in different engines is an important aspect of this dissertation, and the corresponding contributions to this aspect will be presented in Chapter 5.

1.2 Research Questions and Outline

In this dissertation, three primary research questions are investigated.

- How to capture realistic pedestrian movements before and during street crossings?
- How to simulate realistic body movements for pedestrian agents?
- How to visualize pedestrian movements inside ADAS environments?

The contributions to each of these research questions are gathered in their own chapter. In Chapter 3, the focus is on the first research question, presenting different novel paradigms of capturing pedestrian behavior and motion simultaneously. Both, real capturing and virtual capturing

experiments and their results are presented. In Chapter 4, different simulation approaches for the data-driven generation of pedestrian animation in pre-crash simulations are presented, targeting the second research question. In Chapter 5, the MOSIM Framework for motion exchange and an online retargeting approach are presented, targeting the third research question. For each research question, the related work is quite different. Consequently, the related scientific work for each topic is presented at the beginning of each chapter separately. The fundamental background information on autonomous driving, pedestrian simulation, animation, and motion capturing is outlined in Chapter 2. In Chapter 6, the combination of all individual contributions is discussed, and future work for the targeted research questions will be proposed.

Subsequently, my individual research contributions in terms of publications, open-source projects, collaborative research projects, and supervised bachelor and master theses are presented.

1.3 Contributions

Some of my work, particularly for the first two challenges (Chapter 3 and 4), was published and/or presented at peer-reviewed venues. Most of my contributions for the third challenge (Chapter 5) have not yet been disseminated at a scientific venue. However, the MOSIM Framework, which was co-developed by me, is published as open-source software on GitHub and was used by different companies.

1.3.1 Publications

The following contains a comprehensive overview of the publications, presentations, and corresponding contributions in chronological order.

1. Janis Sprenger, Helena Kilger, Christian Müller, Philipp Slusallek, and Sarah Malone. "Capturing Subtle Motion Differences of Pedestrian Street Crossings". In: *Proceedings of the 32nd International Conference on Computer Animation and Social Agents*. CASA'19. Paris, France: Association for Computing Machinery, 2019, pp. 29–32. `https://doi.org/10.1145/3328756.3328776` [Spr+19a]
   Contributions: J.S. and H.K. planned, conducted, and evaluated the study under the supervision of the other authors. J.S. developed the capturing and evaluation software and wrote the manuscript, including feedback from the other authors.
   Usage: Section 3.2

2. Janis Sprenger, Han Du, Noshaba Cheema, Erik Herrmann, Klaus Fischer, and Philipp Slusallek. "Learning a Continuous Control of Motion Style from Natural Examples". In: *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games*. MIG'19. Newcastle upon Tyne, United Kingdom: Association for Computing Machinery, 2019 `https://doi.org/10.1145/3359566.3360082` [Spr+19b]

Contributions: J.S. implemented the approach during his master thesis under the supervision of K.F. and P.S.; H.D. and E.H. supported the development. N.C. provided the neural network utilized for annotation. J.S. drafted the manuscript and received feedback from the other authors.
Usage: Section 4.2

3. Janis Sprenger, Han Du, Noshaba Cheema, and Klaus Fischer. "Variational Interpolating Neural Networks for Locomotion Synthesis". In: *Advances in Transdisciplinary Engineering 11: DHM 2020*. August 2020, pp. 71–81. http://dx.doi.org/10.3233/ATDE200011 [Spr+20]
Contributions: J.S. and H.D. derived the concept under the supervision of K.F.; J.S. implemented and evaluated the concept. J.S. drafted the manuscript with the support of the other authors.
Usage: Section 4.3

4. Lorena Hell, Janis Sprenger, Matthias Klusch, Yoshiyuki Kobayashi, and Christian Müller. "Pedestrian Behavior in Japan and Germany: A Review". In: *2021 IEEE Intelligent Vehicles Symposium (IV)*. 2021, pp. 1529–1536 © 2021 IEEE. https://doi.org/10.1109/IV48863.2021.9575809 [Hel+21]
Contributions: L.H. and J.S. contributed equally to this work, gathering information on the different topics. M.K. coordinated the draft. Y.K. provided feedback and validation regarding the Japanese background. C.M. provided feedback and coordinated the project.
Usage: Section 3.1

5. Janis Sprenger, Lorena Hell, and Matthias Klusch. *Motion Capturing in Large-Scale VR Environments using Consumer Hardware*. Extended abstract presented at the 17th International Symposium of 3-D Analysis of Human Movement. Tokyo, July 2022. [SHK22]
Contributions: J.S. wrote the extended abstract under the supervision of M.K.; J.S. and L.H. prepared and conducted the user study. J.S. implemented the processing and experiment. The presentation received the Best Student Presentation Award.
Usage: Part of Section 3.3.

6. Janis Sprenger, Lorena Hell, Matthias Klusch, Yoshiyuki Kobayashi, Shoma Kudo, and Christian Müller. "Cross-Cultural Behavior Analysis of Street-Crossing Pedestrians in Japan and Germany". In: *2023 IEEE Intelligent Vehicles Symposium (IV)*. 2023, pp. 1–7. © 2023 IEEE. https://doi.org/10.1109/IV55152.2023.10186635 [Spr+23]
Contribution: J.S. and L.H. derived the experimental design, conducted the experiment in Germany, and performed the evaluation under the supervision of M.K. and C.M.; J.S. designed and implemented the experimental software and data processing with the support of colleagues and students. J.S., Y.K., S.K., and others from the National Institute of Advanced Industrial Science and Technology (AIST) translated the text into Japanese and prepared and conducted the experiment in Japan. Y.K. computed the cluster analysis. J.S. and M.K. wrote

the manuscript with the support and feedback of the other authors. The publication was resubmitted to the Computer Science in Cars Symposium 2023 as a short paper, where it received the Best Presentation Award (Short Papers).
Usage: Section 3.3

7. Jan Bohnerth, Lukas Brostek, Klaus Fischer, Janis Sprenger. *Towards Recording Child-Pedestrian Crossing Behavior in Virtual Traffic Scenarios*. Extended abstract presented at the Computer Science in Cars Symposium (CSCS) 2023. December 2023, Darmstadt.
Contributions: J.B. and J.S. implemented, conducted, and evaluated the experiment. L.B. and K.F. provided conceptual input during the preparation of the experiment. J.S. wrote the manuscript. The other authors supported the writing and provided feedback. The work was peer-reviewed and presented at the conference. It received the Second-Best Presentation Award (Short Papers).
Usage: Section 3.4.

8. Ulan Akmatbekov, Janis Sprenger, Rui Xu, Philipp Slusallek. "Simulating Body Movements for Multiple Agent Avoidance". In *2025 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 2025. © 2025 IEEE. [Akm+25]
Contributions: U.A. implemented the approach and performed the evaluation in the scope of his master thesis. J.S. was the students' mentor and wrote the manuscript. R.X. provided feedback on the reinforcement learning configurations. P.S. supervised the thesis and provided conceptual and editorial feedback.
Usage: Section 4.5.

9. Janis Sprenger, André Antakli, Klaus Fischer. "The MOSIM Framework: Simulating Smart Workers in the Industrial Metaverse". In *IEEE Conference on Industrial Cyber-Physical Systems (2025)*. 2025. submitted.
Contributions: J.S. and A.A. wrote the manuscript, while K.F. provided feedback and supervision. J.S.'s focus was on MOSIM and motion simulation. A.A.'s focus was on the AJAN agent simulation framework.
Usage: Chapter 5

I contributed to several other publications that influenced this dissertation but are not directly utilized in this work. The respective publications are listed below:

1. Atanas Poibrenski, Janis Sprenger, and Christian Müller. "Towards a methodology for training with synthetic data on the example of pedestrian detection in a frame-by-frame semantic segmentation task". In: *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*. SEFAIS '18. Gothenburg, Sweden: Association for Computing Machinery, 2018, pp. 31–34. `http://dx.doi.org/10.1145/3194085.3194093` [PSM18]

2. Nico Herbig, Frederik Wiehr, Atanas Poibrenski, Janis Sprenger, and Christian Müller. "How machine perception relates to human perception: visual saliency and distance in a frame-by-frame semantic segmentation task for highly/fully automated driving". In: *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*. SEFAIS '18. Gothenburg, Sweden: Association for Computing Machinery, 2018, pp. 6–10. http://dx.doi.org/10.1145/3194085.3194092 [Her+18]

3. Noshaba Cheema, Somayeh Hosseini, Janis Sprenger, Erik Herrmann, Han Du, Klaus Fischer, and Philipp Slusallek. "Dilated Temporal Fully-Convolutional Network for Semantic Segmentation of Motion Capture Data". In: *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation - Posters*. Ed. by Melina Skouras. The Eurographics Association, 2018. http://dx.doi.org/10.2312/sca.20181185 [Nos+18]

4. Felix Gaisbauer, Jannes Lehwald, Janis Sprenger, and Enrico Rukzio. "Natural Posture Blending Using Deep Neural Networks". In: *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games*. MIG '19. Newcastle upon Tyne, United Kingdom: Association for Computing Machinery, 2019. http://dx.doi.org/10.1145/3359566.3360052 [Gai+19]

5. Tim Dahmen, Patrick Trampert, Faysal Boughorbel, Janis Sprenger, Matthias Klusch, Klaus Fischer, Christian Kübel, and Philipp Slusallek. "Digital reality: a model-based approach to supervised learning from synthetic data". In: *AI Perspectives* 1.1 (2019), p. 2. *AI Perspectives 1, 1 (12 2019)*. http://dx.doi.org/10.1186/s42467-019-0002-0 [Dah+19]

6. Han Du, Erik Herrmann, Janis Sprenger, Klaus Fischer, and Philipp Slusallek. "Stylistic Locomotion Modeling and Synthesis using Variational Generative Models". In: *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games*. MIG '19. Newcastle upon Tyne, United Kingdom: Association for Computing Machinery, 2019. http://dx.doi.org/10.1145/3359566.3360083 [Du+19]

7. Andre Antakli, Torsten Spieldenner, Dmitri Rubinstein, Daniel Spieldenner, Erik Herrmann, Janis Sprenger, and Ingo Zinnikus. "Agent-based Web Supported Simulation of Human-robot Collaboration". In: *Proceedings of the 15th International Conference on Web Information Systems and Technologies - WEBIST*. INSTICC. SciTePress, 2019, pp. 88–99. http://dx.doi.org/10.5220/0008163000880099 [Ant+19]

8. Erik Herrmann, Han Du, André Antakli, Dmitri Rubinstein, René Schubotz, Janis Sprenger, Somayeh Hosseini, Noshaba Cheema, Ingo Zinnikus, Martin Manns, Klaus Fischer, and Philipp Slusallek. "Motion Data and Model Management for Applied Statistical Motion Synthesis". In: *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*. Ed. by Marco Agus, Massimiliano Corsini, and Ruggero Pintus. The Eurographics Association, 2019. http://dx.doi.org/10.2312/stag.20191366 [Her+19b]

9. Erik Herrmann, Han Du, Noshaba Cheema, Janis Sprenger, Somayeh Hosseini, Klaus Fischer, and Philipp Slusallek. "Adaptive gaussian mixture trajectory model for physical model

control using motion capture data". In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '19. Montreal, Quebec, Canada: Association for Computing Machinery, 2019. http://dx.doi.org/10.1145/3306131.3317027 [Her+19a]

10. Chi Zhang, Janis Sprenger, Zhongjun Ni, Christian Berger. "Predicting and Analyzing Pedestrian Crossing Behavior at Unsignalized Crossings". In: *2024 IEEE Intelligent Vehicles Symposium (IV)*. 2024, pp. 674–681. https://doi.org/10.48550/arXiv.2404.09574 [Zha+24a]

11. Chi Zhang, Janis Sprenger, Zhongjun Ni, Christian Berger. "Predicting Pedestrian Crossing Behavior in Germany and Japan: Insights into Model Transferability". In: *IEEE Transactions on Intelligent Vehicles (2024)*, pp. 1-16 http://dx.doi.org/10.1109/TIV.2024.3506727 [Zha+24b]

1.3.2 Open-Source Projects

While the MOSIM and AIToC research projects generated incredible momentum for the simulation of manufacturing workers, most of our research was not published at scientific venues. However, the MOSIM Framework and the corresponding tools are published as open-source projects and are still actively maintained and utilized in research. They can be found in Github (https://github.com/dfki-asr/MOSIM). I contributed particularly to the retargeting, motion transfer, constraint definitions, deployment and repository structure, and communication with the behavior system. The project website (mosim.eu) was lost, and access to the domain is not recommended.

We started to publish the data captured in the virtual reality experiments [Spr+23] following data-privacy guidelines for others to be utilized [Zha+24a; Zha+24b] in a GitHub repository https://github.com/dfki-asr/PedestrianData.

1.3.3 Collaborative Research Projects

During my research work, I have participated in 12 publicly and industry-funded collaborative research projects on national and European levels, primarily on pedestrian simulation and industrial digital twins. Of these 12 projects, I have acquired six projects myself, closed the acquisition of one, and participated in acquiring two others.

Research Projects on Pedestrian Simulation

The REACT project (2017 - 2020) was a BMBF (Federal Ministry of Education and Research) funded corridor project at the German Research Center for Artificial Intelligence (DFKI). The main topic of the research project was to create models and a simulation environment for simulating pedestrian behavior in critical traffic situations for autonomous driving. My team was responsible for the work package on pedestrian motion simulation, and my early work on motion

capturing (Section 3.2, [Spr+19a]) and style simulation (Section 4.2, [Spr+19b]) was funded by REACT.

The CrossCDR project (2020 - 2023) extended the REACT project on a cross-cultural pedestrian behavior analysis in Germany and Japan. Together with my colleagues and colleagues from the AIST at Kashiwa-no-ha in Japan, we derived the virtual reality capturing paradigm to capture and analyze the behavioral and motion data of 120 participants in Saarbrücken, Germany, and Tokyo, Japan. I led the implementation and provided most of the code of the virtual reality (VR) environment and the evaluation software with additional help from Nils Lipp, Ulan Akmatbekov, Niklas Braun, Jan Bohnerth, André Antakli, and Igor Vozniak. The study was primarily derived with Lorena Hell and Matthias Klusch with additional input from Yoshiyuki Kobayashi for Japan. I conducted the study together with Lorena, André, Nils, and several research assistants and colleagues in Germany. I adjusted and prepared the experiment in Japan with the exceptional support of Saori Sampai. I conducted the study with André, Yoshiyuki, Shoma Kudo, and Saori. The statistical comparison and evaluation were performed together with Lorena, Annika Kölsch, and Vanessa Skrobisz. The cross-cultural analysis [Hel+21] and the study's results [Spr+23] were both published and are presented in Chapter 3.

In Momentum (2022 - 2025), we are extending our work on pedestrian simulation to train autonomous vehicles in critical situations. For this, my primary focus is on motion capturing and simulation in these critical scenarios. We are utilizing the data captured in CrossCDR for further evaluations [Zha+24a] and work towards simulation models taking into consideration the environment, other agents (Section 4.4) and their intention.

Within the context of our corridor projects, I acquired and conducted two industry transfer projects with ZF Friedrichshafen AG, Germany's biggest personal vehicle Tier 1 supplier. In the first project (2021), we showed the feasibility of transferring animations into the CARLA driving simulator [Dos+17]. In the second project (2022), we integrated neural network-driven animation models within CARLA and with the pedestrian agent model from CogniBit GmbH.

Together with CogniBit, I acquired the feasibility study ChildPed (2023) funded by the Central Innovation Programm for Small and Medium-Sized Enterprises (ZIM), in which we evaluated the feasibility to utilize our virtual reality capture environment developed in the context of CrossCDR for pedestrian experiments with elementary and pre-school children (Section 3.4). I derived the experiment together with Jan Bohnerth and Lukas Brostek, conducted the study with Jan in Saarbrücken, and performed the evaluation. The results are presented in Section 3.4.

Research Projects on Industrial Digital Twins of Factory Workers

In addition to the pedestrian traffic domain, I worked on several research projects on the simulation of industrial digital twins of factory workers in manufacturing environments.

Within the MOSIM project (2019 - 2021), an ITEA3 (Information Technology for European Advancement) project with several European partners, we developed the MOSIM Framework, an open

modular framework for efficient and interactive simulation and analysis of realistic human motions for professional applications. While the core idea of the MOSIM Framework was derived by Felix Gaisbauer and Philipp Agethen (Daimler), the framework was primarily developed by Felix and myself with the support of different partners and students. My focus was on the implementation of the motion transfer and retargeting approach. The MOSIM Framework and the subsequent extensions are presented in Chapter 4.

In the follow-up project AIToC (2021 - 2024), an ITEA4 project with several European partners, we continued the research with a stronger focus on behavior modeling and the automatic derivation of worker instructions from construction plans. Within the project, I was responsible for the MOSIM Framework, its extensions, and improvements. I restructured the GitHub project to improve its usability, developed several minor improvements, established the constraint chaining approach to move the handling of geometric constraints from the behavior to the motion models, and developed several modular motion units utilized in the final demonstrator.

In the context of AIToC, I acquired an additional industry transfer project with Daimler Busses GmbH to improve grasping motion simulation within the MOSIM Framework. The results were integrated into the final demonstration of AIToC and are published in the GitHub repositories.

In 2023, I acquired the TwinMaP Project (2024 - 2026), funded by the Federal Ministry for Economic Affairs and Climate Action (BMWK), as a follow-up project to AIToC. In this project, we aim to automatize the generation of modular motion units further based on weakly annotated motion capture data and aim to integrate digital twins of machines and tools into an industrial metaverse.

Other Research Projects

In the KAI project (2020 - 2024), which I partially acquired, we investigated the product development process of personal vehicles and the potential to automatize the generation of new interior concepts. We investigated the prediction of posture preference ratings and postures from vehicle descriptions, the description of semantic information within the vehicle and its rating, and the optimization of vehicle configurations. The concepts investigated in this project were partially applied for the optimization processes within the processing of pedestrian motion capture data in the context of this dissertation.

In December 2023, I acquired another research project within the context of the Intel-Saarland cooperation on the “Future of Graphics and Media”, in which we will focus on the network transfer of motion and the compensation of latency when utilized in low-latency applications like close interactions within full-immersive metaverse applications (2024 - 2026).

1.3.4 Supervisions and Teaching

I supervised four internships relevant to this thesis. Two internships (Helena Kilger, Sina Kling) were conducted in collaboration with Dr. Sarah Malone (Saarland University) in traffic psychology,

supporting the experiments in real environments (Section 3.2). Another internship (Niklas Braun) in media informatics (Saarland University) focused on pedestrian simulation within CARLA (Section 5.3). Another internship in computer science (Jan Bohnerth) from the University of Applied Sciences Saarland (HTW Saar) focused on retargeting (Section 5.2.5).

I supervised several bachelor and master students who investigated specific research questions under the umbrella of this dissertation. The results are utilized in Section 4.4, Section 4.5, and Section 5.2.5 and are attributed accordingly. The following theses resulted from this supervision:

1. Jan Bohnerth, Scaling Constraints Instead of Motion - Towards a Scale-Aware Motion Transfer Function in the Motion Simulation Framework MOSIM, Sept. 2021 (Bachelor Thesis HTW Saar) [Boh21]
2. Rolando Morales Suárez, Natural and Accurate Stair Walking Synthesis for Character Control using Environment Information, Okt. 2021 (Master Thesis Saarland University) [Suá21]
3. Chirag Bhuvaneshwara, Simulating Fast-Movements with Mixture-of-Expert Models: An Interactive Boxing Controller, Nov. 2021 (Master Thesis Saarland University) [Bhu21]
4. Jan Bohnerth, Towards VR Avatar Visualization with the Help of Conditional Variational Autoencoders and Recurrent Neural Networks, Okt. 2023 (Master Thesis HTW Saar) [Boh23]
5. Ulan Akmatbekov, Modeling Multiple Avatar Avoidance Movements, April 2024 (Master Thesis Saarland University) [Akm24]

Chapter 2
Fundamentals

This chapter describes fundamental information relevant to all subsequently targeted research questions. The relevant scientific work is not summarized here but in each chapter separately.

2.1 Advanced Driver Assistance Systems (ADAS)

Advanced driver assistance systems are an important step towards autonomous driving. The Society of Automotive Engineers (SAE) defines six levels of automation [Soc21]:

- Level 0: No Driving Automation
- Level 1: Driver Assistance
- Level 2: Partial Driving Automation
- Level 3: Conditional Driving Automation
- Level 4: High Driving Automation
- Level 5: Full Driving Automation

These levels are defined by the vehicle's performance in dynamic driving tasks inside a manufacturer-defined operational design domain. However, they do not specify any further requirements regarding pedestrian safety.

Most assistant systems (steering assistant, lane-keeping assistant, etc.) are primarily relevant when there is still an active driver in the vehicle (levels 1-3). However, their core functionality (e.g., steering, keeping within the lanes) is still required for the higher levels. Pedestrian detection, for example, is considered by the American Automotive Association (AAA) as a relevant aspect of ADAS [Ame19], being deployed in different systems for collision mitigation [CP11]. However, many other systems interact with pedestrians, including the emergency braking assistant, parking assistant, and longitudinal and lateral control in urban traffic to avoid emergency braking maneuvers and enable a smooth ride.

Euro NCAP (European New Car Assessment Program) is Europe's most important testing framework for evaluating the performance of new vehicles before they enter the market. There is a very

strong focus on autonomous emergency braking with several different testing scenarios that are configured to generate a collision if there is no breaking maneuver [Eur23]. These scenarios are:

1. Farside Adult: An adult approaching from the far side of the road (typically left)
2. Nearside Adult: An adult approaching from the near side of the road (typically right)
3. Nearside Child Obstructed: A child running from the near side of the road starting behind an obstruction (parked vehicle)
4. Longitudinal Adult: An adult pedestrian walking on the road in the same direction as the vehicle.
5. Turning Adult: The car is turning on a crossroad (either left or right), and a pedestrian is crossing the road the car is turning into.
6. Reverse Adult/Child: A child or adult is standing or walking behind the car while the car reverses.

In addition, the assisted driving test procedure [Eur24] considers a scenario in which the pedestrian is walking on the side of the road along the direction of travel and needs to be overtaken by in-lane maneuvering (lateral vehicle motion) without stopping unnecessarily or hitting the pedestrian.

2.2 Simulation Software for ADAS

Different software systems simulate traffic scenarios for autonomous vehicles. Unlike numerical simulations utilized to test the vehicle's material, traffic simulation engines need to simulate the sensor input of the vehicle and the surrounding environment with its behavior.

The available driving simulators differ in their features and focus. Car Maker[2,3] has a strong focus on the actual simulation of a digital twin of the vehicle, allowing the whole range of integrations from MIL (model in the loop), SIL (software in the loop), HIL (hardware in the loop), to VIL (vehicle in the loop). However, the sensor simulation and environment generation are not focused, and the software primarily provides camera data for static geometry. While it is widely used in the industry, it is not freely available for research.

There are several small commercial driving simulators like ReplicaR[4] and Tronis[5], which provide sensor simulation, scenario definitions, and basic pedestrian simulations. However, their feature set and quality are questionable, as not many evaluations are published with these software suites.

The CARLA Simulator [Dos+17], supported by Intel and Toyota, is an open-source driving simulator based on the Unreal game engine with the focus of providing an easy-to-use environment for

[2] https://ipg-automotive.com/de/produkte-loesungen/software/carmaker/
[3] A full list of software can be found in the Appendix A.4.1
[4] https://www.automotive-ai.com
[5] https://twt-innovation.de/produkte/

the training of neural networks for vehicle control. It offers various sensor inputs like LIDAR (light imaging, detection, and ranging), camera, depth sensors, and simulated GPS (Global Positioning System), and hardware – like the actuators (brakes, steering, gearbox, etc.) or the sensors (actual cameras, LIDAR, etc.) [Cur+19] – can be connected via ROS (Robot Operating System). Pedestrian models are available and are controlled via standard gaming animations and path navigation; thus, they do not show any crossing-related movement or cues of their crossing intention. Unlike the simulator AirSim [Sha+18], CARLA is still actively developed. It is utilized intensively in research, with over 5,000 publications citing the original paper. Due to the project's open-source nature, integration of custom pedestrian simulation models is feasible.

The graphics card manufacturer NVIDIA has recently published their own driving simulator based on the NVIDIA Omniverse[6]. It strongly focuses on rendering and the photorealistic generation of camera data, offering both SIL and HIL capabilities and basic pedestrian simulation. Based on Omniverse, NVIDIA provides the driving simulator DriveSim[7], which contains pedestrian models and scenarios. However, the pedestrian animations contain severe artifacts, and their usability remains questionable.

2.3 Software Agents and Multi-Agent Systems

Software agents are computational or robotic entities that perceive and act upon their environment and whose behavior depend at least partially on their own experience [Wei99, p. 1]. They should act independently, flexibly, and rationally in different environments. In multi-agent systems, they should be able to communicate with each other in some form. The simulation of multiple agents can yield "emergent behavior," a more complex behavior than originally modeled due to their interactions [FSS03]. In the pedestrian domain, pedestrian agents are defined in a simplified manner as an entity that can perceive the environment, navigate the scene, and interact with other pedestrians given a set of rules [Kuk+01]. Hence, in past approaches, there was a major focus on deriving these rules. A common approach is to separate the possible decisions on a strategic (global, off-road), a tactical (route choice), and an operational (e.g., selection of velocity) level [Daa04; IN08; PYG09]. For this dissertation, the tactical and operational levels are of interest (see Section 1.1).

Unlike an agent, an "avatar" can be defined as a graphical representation of a human user inside a virtual environment [OCo19]. In this dissertation, a pedestrian agent is considered as the digital entity of the simulated pedestrian, and a character or avatar is its graphical representation (mesh, skeleton, texture, etc.).

[6] https://www.nvidia.com/de-de/omniverse/
[7] https://developer.nvidia.com/drive

2.4 Frameworks for Pedestrian Simulation

Crowd simulation is a well-established field in research [Yan+20], and multiple commercial tools are available. However, a framework for the simulations of pedestrians in pre-crash scenarios requires the capability to scale to multiple agents, incorporate traffic-related behavior, and animate the pedestrian's intention accurately enough to be perceived by the virtual autonomous vehicle.

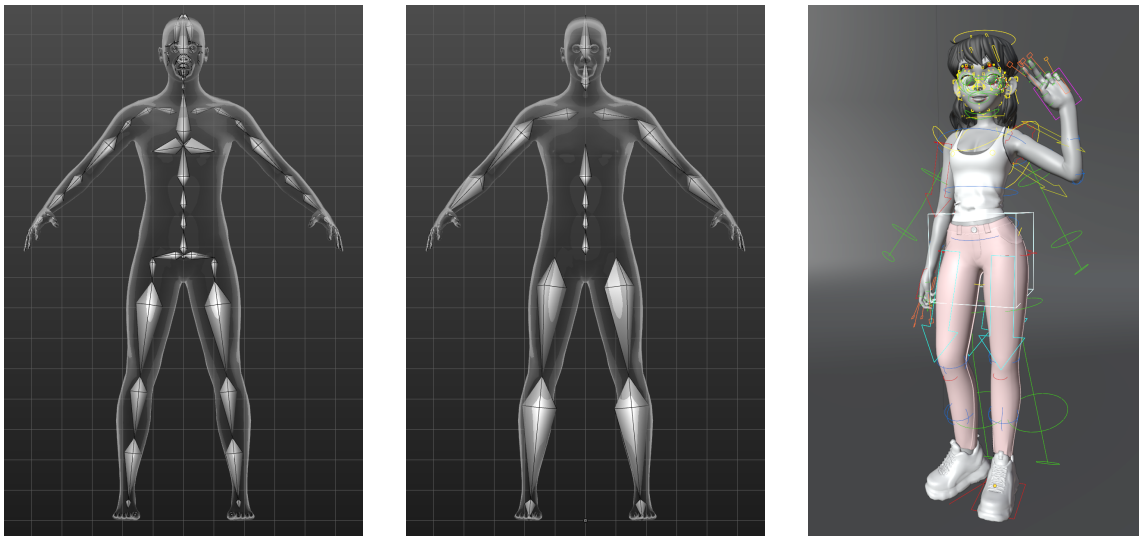
Generic crowd-simulation solutions like Vadere [Kle+19], MassMotion[8], and STEPS[9] focus on the simulation of large crowds and their dynamics. As described in Chapter 1, they operate on a macroscopic scale and are primarily utilized to simulate mass-movement during events and in high-density areas (e.g., airport terminals) or to simulate evacuation scenarios. More specialized solutions like AnyLogic[10] or Sumo [Kra+12] incorporate traffic-specific configurations (like jaywalking, prioritization of sidewalks, gap acceptance, etc.) but fundamentally operate on the same macroscopic level. On a very coarse level, they can simulate the trajectory information for pedestrians. However, they cannot simulate the intricate behavior and motion indicating a specific crossing decision and, thus, are not suited for pre-crash simulations. Driving simulators like CARLA [Dos+17] and Tronis either support the specialized solutions (especially Sumo) or utilize their internal pedestrian simulation. Most of these driving simulators are built on top of a game engine (or provide similar functionality as a game engine), so they allow for more direct control of the pedestrian agents.

To the best of our knowledge, no available solution or approach utilizes the theoretical capability to provide high-fidelity animations of pedestrians to simulate pre-crash situations, displaying their crossing intention accurately enough.

2.5 Virtual Character Animation

Virtual characters inside an engine are displayed with their meshes and textures. Meshes describe geometry as a set of vertices and simple surfaces, while the texture contains information on the reflective properties of the individual surfaces (e.g., color). The skin, clothing, and accessories are usually grouped and considered part of the virtual character. The mesh and the textures can be controlled and simulated directly. In facial animation, for example, specific, pre-defined shapes of the face and their interpolation are utilized to express feelings with facial action units [EF78] and to animate speech with visemes [DPV06]. Simulation models like SMPL (skinned multi-person linear model) [Lop+15] further extend mesh-based animation by encoding a correspondence mesh for different body shapes for the fully articulated body. One benefit of these models is a very high level of control over the visualization of the animation. However, as the mesh is simulated directly, a change of mesh is highly complex and can require the retraining of the complete model. Additionally, adding high-quality clothing and variation in the clothing is challenging.

[8] https://www.arup.com/services/digital/massmotion
[9] https://www.steps.mottmac.com/steps-dynamics
[10] https://www.anylogic.de/



(a) CMU Skeleton **(b)** Game Skeleton **(c)** Control Rig

Figure 2.1: Two different skeleton hierarchies for the same humanoid character and a visualization of a control rig. While each joint needs to be rotated individually in the skeletons, the character pose in the rig is controlled by different high-level control toggles adjusting the end effectors (e.g., hands, feet) and the rotation of in between parts (e.g., upper leg).[11]

A virtual skeleton is commonly used for animation to simplify and exchange animation between different characters. It consists of joints and bones corresponding roughly to the biomechanical skeleton of a human. In a step called "rigging", the influence (weight) of each joint on each vertex is defined. This weight is used to compute the movement of a vertex when a joint is rotated or translated, using weighted blending [LCF00]. Examples of such a virtual skeleton inside a mesh can be found in Figures 2.1a and 2.1b. Figure 2.2 shows the kinematic chain of a skeleton. Individual avatar poses are defined with the kinematic skeleton. A human designer creates individual spatiotemporal key-frames, and via interpolation over time, the virtual human body is animated. In motion capture data, the individual key-frames of the skeleton are captured with a high frequency (e.g., 60 Hz).

To simplify the work of human animators, more complex "control rigs" can be created, which group together different joints and allow for easier posing of characters using inverse kinematics. These control rigs usually do not correspond to the human skeleton anymore but offer different control toggles, e.g., for placing the avatar on the ground, changing the general body orientation, or placing the hand at a specific position. Technically, the rig still utilizes a virtual skeleton to manipulate the mesh. However, the control rig offers a better level of abstraction to the designer and easier character posing. Figure 2.1c shows an example of such a virtual rig.

Configurability and variance of the outer appearance are essential requirements for pedestrian simulation. In addition, motion capture data is utilized as a basis to train simulation models. Hence, this dissertation focuses on skeletal-based animation.

[11] The skeleton characters (Figure 2.1a, 2.1b) were generated with the Makehuman character generator (`http://www.makehumancommunity.org/`). The character "Danny" in Figure 2.1c was created by Ethan Snell.

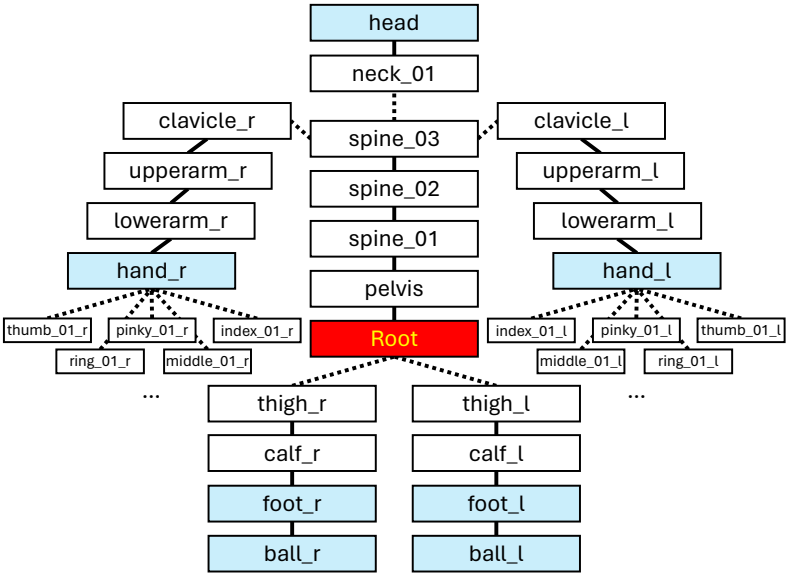
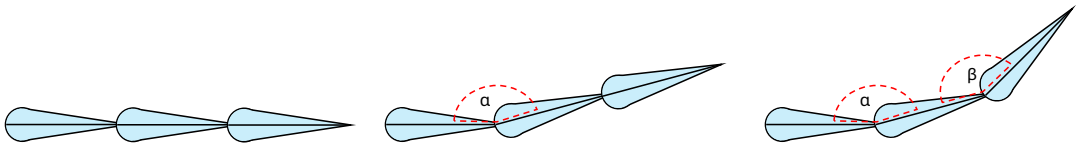


Figure 2.2: The hierarchical graph of the Game Skeleton in Figure 2.1b. The root node is highlighted in red, and typical end-effectors are in blue. Solid connections display a rigid joint connection, and dashed connections are a non-rigid connection of joints.

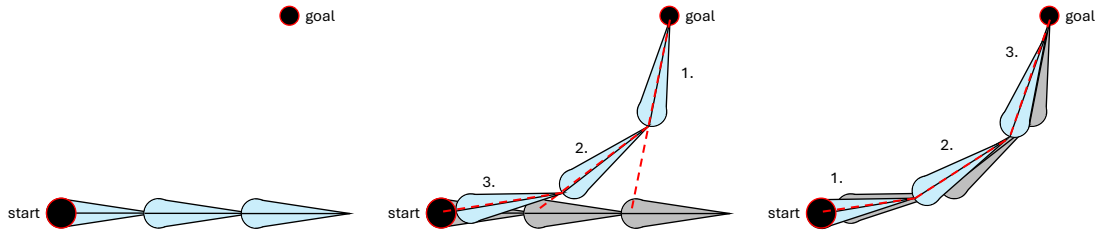
2.6 Forward and Inverse Kinematics

Posing a kinematic character requires to differentiate between forward kinematics (FK) and inverse kinematics (IK). **Forward kinematics (FK)** describes the process of iteratively applying rotations to each joint in a kinematic chain to produce the global Cartesian coordinates of an end effector. The state of each joint j can be defined as a local homogeneous transformation matrix $L_j \in \mathbb{R}^{4x4}$ containing the static translational offset and the animation rotation relative to its parent joint. By subsequently applying the local joint transformations, the global joint transformation G_j can be calculated as $G_j = L_{root} \times ... \times L_{grandparent} \times L_{parent} \times L_j$. A virtual character skeleton can be non-rigid by combining a static translational offset with a flexible offset defined by the animation data. This can compensate for differences between the virtual skeletal representation and the actual physical body and, thus, increase realism. If there is no animation data and the rotations and non-static offsets are zero, the character is in its "zero posture".



(a) Start **(b)** First Rotation **(c)** Second Rotation

Figure 2.3: An example of forward kinematics. The local rotations are applied to the joints starting from the root joint in the depth-first order of the kinematic chain.



(a) Start **(b)** Backward Pass **(c)** Forward Pass

Figure 2.4: An example of the process to achieve an inverse kinematics solution using the FABRIK [AL11] algorithm. First, a backward pass starting at the end effector places the joint at the goal and reorients the parent joints accordingly. In the forward pass, starting from the root, the joints are placed at the start and oriented to their child's joints. Both backward and forward passes are continued until an end criterion is achieved.

Inverse kinematics (IK) is the inverse process in which an end effector should be posed at a specific global position and orientation. To reach this global position and orientation, the rotations for all joints in the chain need to be calculated to position the end effector at the targeted global position if reachable. Unlike forward kinematics, it is an ill-posed problem that can have indefinitely many solutions. The most famous algorithms to solve the IK process in character animation are an iterative Jacobian approach [MM04], a heuristic cyclic coordinate descent [Ken12], or the iterative forward and backward optimization approach FABRIK [AL11] (Forward and Backward Reaching Inverse Kinematics). From the described algorithms, the FABRIK algorithm is the easiest to implement for simple kinematic chains, and an example showing the process can be found in Figure 2.4.

For posing humanoid characters, an IK solver requires additional functionality to optimize the pose for multiple children (e.g., between the pelvis and the legs, between the spine and the shoulders), prevent twisting of bones (rotation along the bone axis of the joint) and maintain biological constraints (e.g., prevent overextension of the knee joint). Professional inverse kinematics solutions contain additional code and operations (e.g., FinalIK[12]) and optimization algorithms with complex constraints [Sta20] to solve these challenges.

During this dissertation, multiple tests with custom IK implementations based on the FABRIK algorithm have been conducted. Ultimately, however, available IK solutions integrated in Unity[13] and Blender[14], as well as FinalIK have been utilized, as they provide a more versatile solution with a more natural result as a self-made algorithm implemented in a reasonable amount of time.

[12] http://root-motion.com/
[13] https://unity.com/
[14] https://www.blender.org/

2.7 Motion Capturing

There are different ways in which the motions of real humans can be captured and transformed into animations of virtual characters. Unlike pose prediction, the goal is not only to detect the pose of the human but to achieve a consistently sized, smooth, and accurate animation with minimal animation artifacts (e.g., foot sliding, foot skating, popping joints, etc.). There are different approaches to motion capturing, and the most common approaches are outlined below:

1. **Monocular Motion Capturing**: Using a single camera, it is possible to estimate the poses of a person in 3D space (e.g., [Meh+20; Shi+20]). While this approach offers the cheapest solution to motion capturing, it usually results in the poorest data quality. Predicting skeletons in a consistent size, matching the right 3D position, and predicting joint positions in the case of self-occlusions is highly challenging. However, due to the advances in the past years, capturing motions with monocular cameras (e.g., inside the smartphone) for character animation with reasonable quality has become a feasible solution. [Shi+20]

2. **Stereo Camera Capturing**: A stereo camera offers additional depth information which improves spatial accuracy and positioning for motion capturing (e.g., [WZC12; Wu+13]). The fundamental challenges (especially self-occlusion), however, remain the same.

3. **Multi-Camera Markerless Capturing**: To perceive the character from all angles, a camera array can be placed around the capturing area (e.g., Captury[15]). Using an extrinsic calibration, the 3D reconstruction can achieve excellent accuracy. Self-occlusion is almost completely eliminated, and additional objects and their shape can be tracked. The existence of clothing can be a benefit and a drawback. While the movement of clothing can be tracked, reconstructed, and utilized for mesh-based animation systems [Agu+08], the accuracy of the skeletal prediction can be reduced. Figure 2.5a shows an example of a multi-camera capturing lab.

4. **Marker-based Capturing**: One of the oldest methods for motion capturing is marker-based motion capturing, in which reflective markers are stuck to specific landmarks on the human body. Multiple cameras observe the capturing area and track the reflective markers in space. The reconstruction software can generate human animation with the highest accuracy in all capturing solutions (e.g., Vicon[16], OptiTrack[17]). An example of the reflectors attached to a human participant can be seen in Figure 2.5b.

5. **IMU-based Capturing**: Instead of using an optical sensor to capture the motion, it is possible to utilize multiple inertia measurement units (IMUs) attached to the body (e.g., XSens[18,19]). Each unit measures its orientation in space and its acceleration. The human body poses can be reconstructed by placing the sensors at specific landmarks and using a calibration

[15] https://captury.com/
[16] https://www.vicon.com/
[17] https://www.optitrack.com/
[18] https://www.movella.com/products/motion-capture/xsens-mvn-awinda
[19] A full list of hardware can be found in Appendix A.4.2

(a) Multi-Camera Lab **(b)** Marker-Based Capturing **(c)** IMU-Based Capturing

Figure 2.5: Examples for different motion capture systems. In Figure 2.5a, one of the users is wearing an additional VR headset that is unrelated to the capturing. In Figure 2.5c, the VR headset and VR trackers are utilized for additional positional guiding.

sequence. IMU-based capturing can generate highly realistic animations, does not suffer from occlusions, and has an almost unlimited capturing area. However, there is no external reference point, so the spatial accuracy is limited, and external objects cannot be accurately tracked without an additional optical system. An example of an IMU-based capturing suit with positional guiding using a Vive VR setup can be seen in Figure 2.5c.

In this dissertation, we primarily utilized IMU-based motion capturing. First, IMU-based motion capturing can be utilized in arbitrarily large and real outdoor street environments. Outdoor capturing is not feasible with multi-camera and marker-based capturing. Marker- and camera-based capturing requires capture studios, which were neither available nor usable for larger spaces. Monocular capturing would only be feasible if the capturing person moves the camera to achieve an occlusion-free view and remains within the specified space (distance, field of view). Second, IMU-based capturing is highly flexible and does not require a specific capturing lab. Any large room (e.g., meeting room, sports court, foyer) can easily be converted into a motion capture lab without a significant setup time. Third, IMU sensors can be worn on top of most clothing without participants having to wear specific clothing (as in the case of multi-camera capturing) or almost no clothing (as in the case of marker-based capturing). Due to these reasons, IMU-based motion capturing was the most reasonable approach for this dissertation.

2.8 Real and Synthetic Data: Domain Shifts

Autonomous vehicles operate in a real environment, but the vehicles' perception and control models should also be trained and tested in a synthetic simulation environment. However, it is typically assumed that the training and testing data for machine learning methods have the

same data distribution. Especially when working with synthetic data, there are differences in the data distribution (domain shifts) between the real and synthetic data. Even with highly realistic rendering (including global illumination as in Nvidia Omniverse, Blender Cycles, or Unreal Engine 5[20]), achieving the same image distribution is hard. Besides the pure rendering, the scene content must match, including the geometry, the dynamics, and the animations. While the former has been focused on significantly, the animations – especially for pedestrians – have not received significant attention yet. For a more comprehensive overview of domain adaptation methods, we refer to respective review articles [Far+21] and datasets [Sun+22].

2.9 Generative Artificial Intelligence and Image Synthesis

Besides rendering images with a classical rasterization or ray-tracing approach, recent advances have been made in generating sensor data with neural networks. Stable diffusion [Rom+22] has shown great performance for static image synthesis in various tasks and has been cited over 8,600 times in just two years. Several text-to-image models are readily available online and have been partially used to generate some figures in this dissertation. OpenAI's Sora[21] has shown new levels of temporal stability for video generation, showing temporally consistent videos with increasingly plausible content. I refer to the review articles for more information on general approaches for neural rendering and generative image models [Tew+20; Tew+22; GG23]. However, they will most likely be outdated within the next few years. Concepts of neural rendering have been applied to autonomous driving as well (e.g., [LLZ23; Che+23]).

While generative artificial intelligence (AI) will have a significant impact on synthetic data generation in the next decade, it is unlikely that it will completely replace classical simulation in the near future. Even models like Sora are still struggling with basic human animations. While the videos look highly realistic at first glance, the human animations contain many errors, from falsely scaled pedestrians, pedestrians vanishing behind objects, and classical animation artifacts like foot sliding, skating, and skipping. On top of that, there is only weak control over what happens explicitly within the scene, complicating the generation of reproducible and precise testing scenarios. The generative AI models utilized to synthesize the videos require a large amount of training data and thus add additional complexity when trying to generate unbiased and localized simulations. Lastly, one of the benefits of classical simulation is the generation of matching multi-sensor simulations (e.g., camera, lidar, radar) with high-quality labels (e.g., pixel-accurate annotations), which is still impossible with black-box image generators.

[20] https://www.unrealengine.com
[21] https://openai.com/sora

Chapter 3
Capturing Pedestrian Motions

It is fundamentally important to understand how pedestrians behave in traffic environments to develop realistic pedestrian agent models. For the animation of these agents, it is particularly important to know how the pedestrians move their bodies to indicate and execute their behavior. While pedestrian behavior, in general, has been studied intensively in traffic psychology (see Section 3.1), the data is usually not captured and reported in a machine-readable way allowing the training of generative simulation models. While typical traffic-psychology measures – like average velocity and the acceptable gap between vehicles for crossing – are intensively studied and can be utilized to configure a conventionally simulated agent, to my knowledge, there have been few studies on the actual physical body motions before entering or during crossing of the road, with the exception of Kalantarov et al. [KRO18].

For a realistic animation of a virtual pedestrian agent, it is essential that the intention (e.g., a person wants to cross, does not want to cross) is accurately visualized through the agent's body postures. Besides an investigation on which foot is utilized to step on the road surface [KRO18], there is no information on the body postures described in the literature. Consequently, we conducted several new studies to capture the behavior and motion of pedestrians crossing the street. To be able to train generative motion models, it is essential to conduct the experiments in such a way, that sufficient training data is generated. We conducted four studies with over 150 participants using motion capturing to gather the required data. In all studies, the full-body motion data of participants was recorded using an IMU-based motion capture suit.

The first study (Section 3.2) was performed in a real street environment on the campus of the Saarland University. Utilizing an IMU-based motion capture suit and a Microsoft Hololens for absolute positioning, the natural body rotations to indicate a crossing decision were observed, recorded, and evaluated. The critical behavior of reeving into a zebra crossing to cross the road was analyzed and compared against non-critical behavior, where participants remained on a single side of the road without a crossing event. The importance of the torso and lower body rotations to differentiate non-critical from critical behavior was identified.

The second study extended the first, investigating crossings on a low-density four-way crossroad. However, the pandemic and the resulting lockdowns in 2020 substantially impacted the traffic environment on campus and the continuation of the second study.

The third study (Section 3.3) was conducted in a large-scale virtual reality (VR) environment to compensate for the shortcomings of real-environment testing. The study focused on the cultural differences between Japanese and German pedestrians. For this, 60 participants were recorded in Germany and 60 in Japan in different street crossing environments (with and without zebra crossing, with and without other pedestrians). The data shows significant differences in the behavior and movement patterns between both groups. To date, the observed behavior has not been reported in literature.

In the fourth study (Section 3.4), the paradigm was extended with more environments (one- and two-lane roads, signalized crossings, crossing behind an obstacle). The behavior of (primary school) children was investigated in these different environments. The results demonstrate the feasibility of conducting crossing studies with children unaccompanied by parents in a safe environment, which would be impossible in real environments.

3.1 Related Work on Pedestrian Behavior, Movements and its Capturing

3.1.1 Different Layers of Pedestrian Behavior

Based on the literature analysis in my Bachelor thesis in psychology [Spr19] and following the software-agent paradigm [Wei99], we propose a conceptual three-layer pedestrian model to structure the literature analysis. On the **Cooperative Layer**, interaction processes between the ego-pedestrian (the simulated pedestrian) and other entities within the traffic environment (other pedestrians, bicyclists, drivers, cars, etc.) are described. On the **Tactical Layer**, the state of the environment influences high-level decisions. These decisions include the route choice, compliance with traffic rules, the accepted risk, or the minimal distance (gap) between cars for a save crossing. The **Operational Layer**, or action layer, describes how the actual ego-pedestrian body is moved to enact the decisions and behaviors of the higher layers, including velocity, gait, attention, and gaze.

All three layers are interconnected: While communication between the ego-pedestrian and other drivers might influence the crossing decision on one hand, it might additionally involve certain actions like raising the arm to acknowledge yielding, which need to be visualized. On the other hand, attention and gaze – on the operational layer – can lead to the perception of a new vehicle occluded before and thus trigger a new deliberation on how the driver might behave in the future and whether a crossing can still be considered safe.

Unlike for city planning and traffic analysis, the question of why and when pedestrians commute is irrelevant for pre-crash simulations and for this dissertation. Thus, literature on this strategic level is not further analyzed, and we refer to respective review articles [IN08].

Cooporative Layer

For this dissertation, two primary interaction partners in traffic are considered: pedestrians, and drivers in their vehicles, particularly pedestrian-pedestrian and pedestrian-driver interactions. Other interactions (e.g., driver-driver, pedestrian-cyclist) are excluded from this review.

For **pedestrian-pedestrian interaction**, different aspects should be considered. An inverse quadratic relation exists between the number of pedestrians and accidents in a given area [EB17]. Crossing in groups can create a feeling of safety, and other pedestrians' behavior is utilized as a cue for their own crossing decision [Ros09]. The likelihood of crossing increases by a factor of $1.5 - 2.5$ if another person has already started to cross [FKK10], even on a red light and in dangerous situations indicated through aborted crossing attempts. The magnitude and importance of this effect, however, is debatable, as it competes with other external (traffic lights, traffic volume, etc.) and internal (social norms, risk disposition, etc.) factors [Ros09; Spr+23]. Other explicit social interactions, like waving or pointing, have not been analyzed by traffic psychology studies but are highly relevant regarding intention estimation and simulation [Mus+21]. Due to preceding social

cues (e.g., waving), an automated vehicle can estimate the interconnection between and thus the increased likelihood of a crossing decision of the pedestrians [Mus+21].

Interaction between **pedestrians and drivers** can occur with different severity, ranging from crossing without interaction to distant interaction, conflicts, and ultimately collisions [Clo+17]: Generally, non-interactive crossing pedestrians, who wait until no vehicles are visible or all vehicles yielded, can be differentiated from interactive crossing pedestrians, who pause, stop mid-way, vary their crossing velocity, or show explicit communication (e.g., gestures) [Oxl+97]. Rasouli et al. [RKT17; RKT18] report 80% of pedestrians looking at the traffic before crossing, with 90% of these cases indicating a potential attempt to establish eye contact. Additionally, 15% of the cases contained more explicit forms of communication (nodding, gestures, etc.). On the other hand, Dey et al. [DT17] report that the intent for crossing was signalized by stepping on the street (61%), trajectory adjustment (20.7%), or waiting for cars to slow down (12.6%). They could only report 2.7% of direct interactions and 3% gestures for acknowledging or appreciation of yielding during and after the crossing. The reported gaze duration to the individual cars was too short for the authors to assume the attempt to establish eye contact. Although the human eye can distinguish small objects at incredible distances under ideal conditions, it is safe to assume that direct eye contact is only possible at short distances. With increasing distance, the driver's behavior is only visible in the car's movements.

Tactical Layer

The tactical level deals with the short term decisions of pedestrian's behavior. This starts with selecting the next intermediate target, the resulting intention to cross, compliance with traffic rules, other risk-related behavior, and ultimately, the decision when and where to initiate a road crossing.

Intermediate Targets. The intermediate targets are the individual traffic facilities (signalized crossings, zebra crossings) or crossing options (mid-block crossing, space between parked vehicles, etc.) [PYG10]. While signalized crossings offer the safest option through decoupling traffic and pedestrian flow, they usually require a certain wait time for the light to switch. The decision of whether to utilize the safe facilities or to attempt a more risky crossing without any facility largely depends on the traffic flow, the distance to the facility, and the expected delay [PYG09; Pap12; CAR15]. For example, pedestrians do not necessarily cross on the shortest (orthogonal) path but tend to frequently cross diagonally at signalized (14 - 17%) and unsignalized (ca. 36%) crossings [TG11]. Human factors appear to have no major influence on the route choice [PLY16], except for age. Due to age-related deficits, elderly pedestrians are reported to prefer signalized crossings [TDC16]. However, detours require a higher effort and due to physical impairments, the majority of elderly pedestrians are reported to cross at their current position, most of all during sparse traffic and good visibility [TDC16].

Compliance. The compliance rate at signalized crossings is about 60% [GHT03; Dom+15; OA15], which varies with traffic volume [SA03; DOA15], consecutive crossings [Lan+11], wait time [Bro+13; Lip+13] and between cultural backgrounds [Hel+21]. Young adults [Bro+13] and male adults [GHT03;

Gra07; TG11] tend to violate traffic rules more often. Tom et al. [TG11] report that while all observed pedestrians at unsignalized zebra crossings tend to cross within five meters, only 86.5% crossed exactly within the zebra crossing. Regarding overpasses, Demiroz et al. [DOA15] report that time-saving was the primary reason for jaywalking and not using a longer overpass. Brosseau et al. [Bro+13] report an increasing probability of illegal crossings with an increased wait time at a signalized crossing. Countdown devices indicating the remaining time of the green phase increased violations due to underestimating the crossing duration, while countdown devices during the red phase decreased violations [Bro+13].

Risk-Related Behavior. Besides non-compliance, there are other risk-related behaviors. Generally, these street crossing behavior variants can be considered safe: straight crossing inside a marked crosswalk, stopping at the curb and looking (left-right-left) before initiating the crossing, deciding to cross with a sufficient safety margin to the approaching vehicles, and walking at a steady pace without running (diminished ability to adjust maximum velocity [ZK03; Gra07; SSK09]). Dunbar et al. [Dun12] analyzed a British accident analysis database and found the relative risk of accidents shifts between the near and far sides of the road with age. The risk of a near-side accident is about twice as high for children aged 10-14 than for far-side accidents. This relation decreases over age to a factor of 1.5 for elderly pedestrians. Interestingly, however, young children under 9 have a similar relative risk compared to these elderly people. The authors assume that younger children are primarily accompanied by adults. Charron et al. [CFG12] report high observation failure for children crossing in a virtual environment (mean age 10.4 years). Meir et al. [MPO13] report that children aged 7-10 refer to zebra crossings more often as a salient cue than older children or adults. Different studies analyzed the effect of smartphone usage but consistently found a higher number of missed crossing opportunities rather than a higher amount of collisions [Nei+10; Nei+11; Sch+12].

Gap Selection and Acceptance. There are two mechanisms utilized by pedestrians to estimate whether a gap in traffic is safe to cross. The distance-based heuristic relies on the distance of the approaching vehicles and requires minimal information (e.g., a single glance). The time-based heuristic estimates the time-of-arrival of approaching vehicles and thus considers the velocity of the vehicle. There is evidence that with an increasing velocity of the approaching vehicles, the judgmental precision is reduced [Con+98]. Young children (below 7 years) [SGS08] as well as old-old[22] pedestrians (75+ years) [Oxl+97; LC07] rely more on the distance-based heuristic, and this effect is more pronounced under difficult situations [Oxl+97]. Hence, it can be assumed that the distance-based heuristic requires less effort than the time-based heuristic. Conelly et al. [Con+98] found that younger children (5-6 years) tend to make more conservative crossing decisions than older children (8-9 years). On the other hand, the crossing decision does not vary in elderly pedestrians up to a high age. Only the group of old-old pedestrians (75+ years) show significant differences from other adults. Unfortunately, the results are inconclusive. While Oxley et al. [Oxl+05] and Holland et al. [HH10] report reduced safety margins and crossing on smaller

[22]There is a differentiation of young-old (60-69) and old-old (75+) pedestrians in literature due to significant differences frequently found between both groups

gaps for elderly pedestrians, Lobjois et al. [LC07; LC09; LBC13] report safer crossing decisions of elderly to compensate for their reduced motor function.

Operational Layer

The behavior and intention become obvious only by analysis of the actual movements of the human body. While trajectory prediction is primarily based on the past observed trajectory of the pedestrian as a whole, intention prediction usually requires more information [ZB23]. Besides the communication with others [KRT20], this includes the full body postures or motion states [Zha+23a]. While there is limited information on the actual body movements and few datasets of sufficient quality for the training and configuration of generative motion synthesis models, different studies analyze the velocity and gait.

Gaze. Pedestrians gaze is usually directed on the approaching vehicles, traffic controls, and the ground, particularly the curb [GHT03; Dom+15]. In cases where there is no dedicated pedestrian crossing phase, e.g., on unsignalized zebra crossings [GHT03] and at red-light crossing violations [TG11], approaching vehicles are most looked at. The ground and curb are primarily focused on to avoid missteps and falls, especially with elderly pedestrians [TDC16]. Apart from that, the gaze behavior of elderly pedestrians does not differ from other adult age groups [TDC16]. While Dommes et al. [Dom+15] report a time-continuous gaze with pedestrians fixating on objects in similar proportions before and during the crossing, Geruschat et al. [GHT03] report significant differences between both phases.

Velocity. With increasing age, the locomotion velocity first increases until the age of 20 and then gradually decreases again [EH77]. A higher velocity results in more risky crossings, as the potential to increase the own velocity further diminishes. Thus, the potential to adapt the behavior to a new situation is reduced. The absolute crossing velocity depends on factors like age, group size, weather, safety margin, type of crossing, etc. [IN08]. The absolute velocities reported during crossings differ between studies. Demiroz et al. [DOA15] reports an average crossing speed of 1.21 m/s (max 1.74 m/s), with a higher crossing velocity (1.67 m/s) on roads with a high-speed limit (70 km/h) and lower crossing velocity (1.00 m/s) on roads with a lower speed limit (50 km/h). On the other hand, Schwebel et al. [SPS09] report a crossing velocity of 1.34 m/s (speed limit: 50 km/h). Dommes et al. [DCO13] report a decrease in crossing velocity from 1.46 m/s (28.3 years) to 1.08 m/s (76.9 years) with increasing age. Connelly et al. [Con+98] report younger children (5-6 years old) to cross slower (1.27 m/s) than older children (11-12 years old; 1.44 m/s).

Gait. The gait defines the way people move and depends primarily on velocity. Walking and fast walking are in the range of 1 - 2 m/s, jogging in the range of 2 - 4 m/s and running in the range of 4 - 8 m/s [EH77]. While the regular walking gait contains a succession of left and right stances connected with a swing phase, jogging, and running have additional fly phases in which neither foot is connected to the ground. Tom et al. [TG11] report primarily regular walking (94.5%) when approaching the curb, and within half a meter of the curb 18.2% of pedestrians slow down or stop, 1.8% run, and 80.1% continue to walk. On the other hand, Dommes et al. [Dom+15] report

22.7% pedestrians running before entering the street and 6.9% running on the road's pavement. Increasing age is associated with a shorter stride length, a wider standing width, and a longer double stance phase [Sal10]. While still able to "run", the velocity in all gait types decreases significantly [EH77]. Besides the gaze, the full body movements are an equally important indicator of the intended movement direction. While there is some analysis of the foot movement before crossing [KRO18], the effect of full body movements is primarily analyzed for the ability of the driver to detect a pedestrian [Tyr+09].

Human Factors in Pedestrian Behavior

As outlined above, there are only two human factors that are consistently influencing crossing behavior: age and gender. Men are consistently showing more risky behaviors, more traffic violations [TG11; Coh+13], higher velocities [DOA15], and more running on signalized crossings, primarily due to red light violations [TG11]. There are significant differences between regular adult, (young) children and old-old pedestrians (over 75 years). The differences are not only physical (e.g., different crossing velocities [LC07; LC09; DCO13], different vantage points), but also cognitive (e.g., different heuristics to estimate a safe crossing) [Vin81] and self-conscious (e.g., overestimation of walking speed) [DCO13].

3.1.2 Available Datasets

There are different datasets available that can be utilized to train and test of pedestrian prediction tasks like:

1. Pedestrian detection: Where is the pedestrian?
2. Pedestrian path prediction: Where was the pedestrian in the past, and where will she be in the future?
3. Pedestrian intention prediction: Does the pedestrian want to cross?

Datasets often contain information allowing multiple tasks. Although some of these datasets contain ground-truth data for pose prediction, the quality of the pose information is not suited to train generative motion models. While generative motion models require temporally dense information on the joint positions in 3D Cartesian space, datasets contain temporally sparse annotation of joint projections in the 2D camera space [Fra+12]. Nevertheless, these datasets provide a great theoretical basis for this work. The available datasets can be separated into three categories. First, general pedestrian trajectory datasets; second, general datasets for autonomous driving; and third, specific datasets for different pedestrian prediction tasks, containing data from the vehicle's perspective in actual traffic scenarios.

General Pedestrian Tracking Datasets. Although these datasets are frequently utilized for path prediction, they are usually not captured for actual traffic scenarios, and their perspective does not

allow for any detailed movement analysis of the pedestrians. In some cases, the pedestrians are perceived at eye level, but in non-traffic related environments (e.g., [DT05; ELV07]). In other cases, the pedestrians are observed from a top-down perspective (e.g., [Pel+09]). While the top-down perspective is ideal for pedestrian trajectory tracking and prediction, it is less ideal for capturing the subtle motions before crossing (e.g., gaze, upper- and lower-body rotation, leg movement, etc.).

General Autonomous Driving Datasets. General datasets for autonomous driving, like the famous KITTI dataset [GLU12], contain intrinsic information on the vehicle (e.g. GPS, IMU, etc.), object detection annotation, dense pixel annotations (semantic segmentation, e.g. [BFC09; Cor+16]), or LIDAR and radar sensor information [Cae+20]. There have been attempts to create digital clones (e.g., [Gai+16]), but the animation of pedestrians is still an unsolved challenge. Many datasets focus more on representing a diverse set of driving situations and environments (e.g., [Yu+20; Cae+20]) to enable the development of robust object detection, object tracking, self-localization, and segmentation approaches. For image-based tasks like segmentation or detection, annotation is temporarily sparse (one frame per sequence [Yu+20], to 5-6 frames per second (fps) [Cor+16]), but more dense for tracking tasks (30 fps [Yu+20]).

Specific Datasets for Pedestrian Prediction. While general datasets usually do not contain any special information on pedestrians except their bounding box or semantic segmentation, several datasets are specifically constructed to enable the training and evaluation of pedestrian prediction tasks. This includes more detailed annotations (e.g., multi-body segmentation [Li+17]), datasets containing diverse scenes with complex pedestrian movements and interactions (e.g., [Fra+12; KG14; Hwa+15; ZBS17; Bra+19]), and more trajectory information (e.g., [Ma+19]). The JAAD dataset captured by Rasouli and Kotseruba [RKT18] is particularly important, as it does not only contain pedestrian detection and tracking annotations but a time-dense annotation of the behavior (e.g., looking, crossing, waiting, etc.) of both, the driver and the pedestrian.

While all of these datasets are important for their respective application areas, they are suboptimal for the training of generative pedestrian models, as targeted by this dissertation. In particular, the temporally dense pose annotation is missing in almost all cases. Creating valid annotations is labor-intensive and can only be estimated from the camera images. Extracting the motion from existing datasets is almost infeasible. On the other hand, some of the existing datasets contain naturalistic scenes with imprecise annotations. Most of all, the pedestrian's intention is unknown in cases where no crossing event has been recorded. In addition, capturing critical scenarios (near misses, collisions, etc.) is almost impossible in naturalistic environments due to ethical and legal constraints.

3.1.3 Cultural Differences between Germany and Japan

Pedestrian behavior in urban traffic scenarios can significantly differ worldwide, partially due to different rule sets and partially due to social, cultural, and anthropometric [Int10] differences. As traffic law is less specific for pedestrians than vehicles, analyzing the cultural differences between

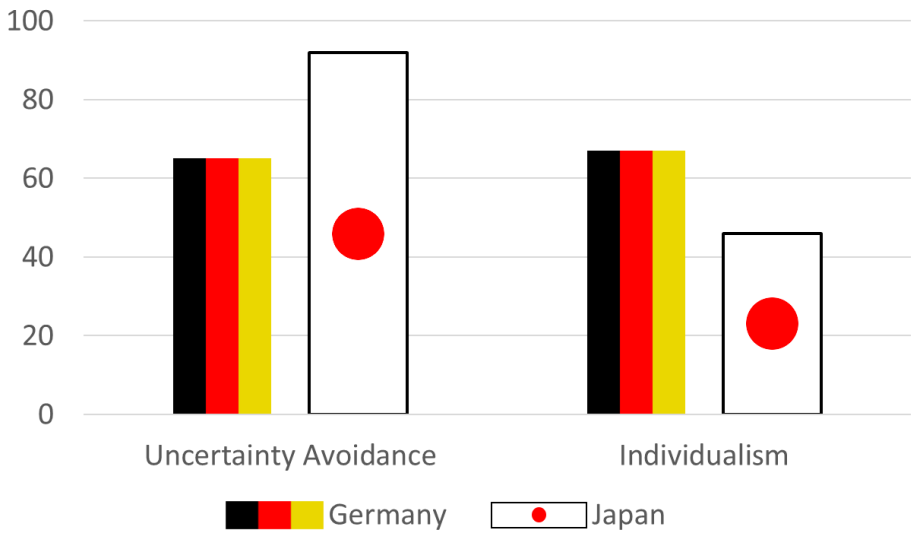


Figure 3.1: Hofstede scores of uncertainty avoidance and individualism for Germany and Japan (cf. [Ins24]). Reprinted with permission from [Hel+21] © 2021 IEEE.

pedestrians becomes more important. Here, the cultural differences are analyzed using Japan and Germany as examples, as both have comparable traffic regulations and are similarly developed countries.

In both countries, walking is considered a relatively popular mode of transportation: 54.5% of Germans report participating in transportation as pedestrians daily, and another 23.9% walk one to three times a week [GHÖ14]. 71.1% of Japanese report walking in the neighborhood for daily errands [Ino+10]. The high level of pedestrian participation in road traffic highlights the high relevance of pedestrian research. Given alternative travel modes available, it suggests a certain sense of safety among pedestrians.

General Differences. A multi-dimensional approach to identify and measure the differences between cultural groups is proposed by Hofstede et al. [HHM05; Hof11]. Two dimensions are particularly important for pedestrian behavior: (i) uncertainty avoidance and (ii) collectivism vs. individualism. Differences in these measures for Japan and Germany are shown in Figure 3.1.

Uncertainty Avoidance. In the case of high uncertainty avoidance, the potential for uncertain situations is reduced by rules, laws, and behavioral norms [Hof11]. This concept can be related to risk avoidance to the extent to which members of a society feel threatened by ambiguous or risky situations with unknown outcomes [MC03]. For uncertainty and risk avoidance, Asian countries, particularly Japan, show higher scores of risk avoidance compared to European countries, and particularly Germany [MC03].

Collectivism. In European individualistic cultures, people tend to consider themselves as distinct individuals with unique characteristics striving for autonomy and independence [HHM05]. Interpersonal relations are largely regulated by individual preferences. In contrast, Asian collectivist cultures interdependence, harmony, and cohesion are encouraged values [Yam01]. Members

of such cultures strongly identify with their in-group as a major source of identity. Although Japan's orientation towards collectivism has gradually declined over the past decades, it is still a fundamental part of its culture [BBU12].

Differences in Traffic Regulations. Many aspects of traffic regulations are similar in Japan and Germany. For example, speed limits in residential (30 km/h) and urban regulated environments (50 km/h) are the same [Nat09; WFF19]. Crossing at a red light and crossing the street without using a crossing facility can be fined (§ 25 StVO and § 121 (1) Road Traffic Act). However, the fine in Germany is significantly lower (5 - 10 €, up to one demerit point in the Driver Qualification Register[23]) [VFR] than in Japan (ca. 20.000 ¥ or 160 € and potential criminal charges) [Gov60]. Pedestrian traffic signals are always on the opposite side of the crossing, showing the red light on the top of a vertical orientation. The green light in Japan is often referred to as "blue", as it has a slightly more pronounced blue tint than in Germany [Nak87]. Additionally, both lights are displaying figurines in standing and walking poses. In Germany, most of the pedestrian signals are operating in an actuated or timed mode [WFF19], while in Japan, 43% of signals are in an adaptive control loop, 48% in fixed-time control, and only 8% are operating in an isolated control [NIO19]. Actuated controls in Japan are used mostly in low pedestrian demand zones and can include elderly-actuated controls [NIO19]. While countdown devices are rarely implemented in Germany, they are quite common in Japan [NIO19]. On the other hand, acoustic devices are mandatory and widely available in Germany [WFF19] and are not yet fully deployed in Japan. In both countries, the minimal green time is chosen to allow pedestrians to cross half the distance [WFF19]. In Japan, the expected number of waiting pedestrians, the flow rate, and the crosswalk width are considered as well [NIO19]. The major difference between both countries: Japanese traffic lights have a constant green mode and a flashing green mode *(PFG – pedestrian flashing green)*, on which a crossing should not be started anymore but finished as soon as possible. German pedestrian lights do not consider such a phase. In Germany, pedestrians receive a head-start compared to turning traffic in many cases [WFF19], but isolated and exclusive phases for pedestrians are rarely found apart from mid-block crossings. An example of the signal phases can be found in Figure 3.2.

One major difference between the two countries is the handedness of traffic. In Germany, traffic is driven on the right side of the road, and in Japan, on the left side of the road. This is not as relevant for pedestrians, as they can approach the road from both sides. Hence, from the perspective of the pedestrians, the road is separated into the near lane (first lane to cross) and the far lane (second lane to cross). To our knowledge, no reports indicate a significant influence of the handedness of traffic on pedestrian behavior or movement.

Compliance. We are not aware of any studies comparing the compliance rate of Japanese and German pedestrians directly [PDS19b]. Studies conducted in both countries separately report lower compliance rates in Germany (61% - 94%) [Lan+11; Lan+16] compared to Japanese pedestrians (93% - 98%) [Sue+13; Pel+17].

[23] "Ein Punkt in Flensburg"

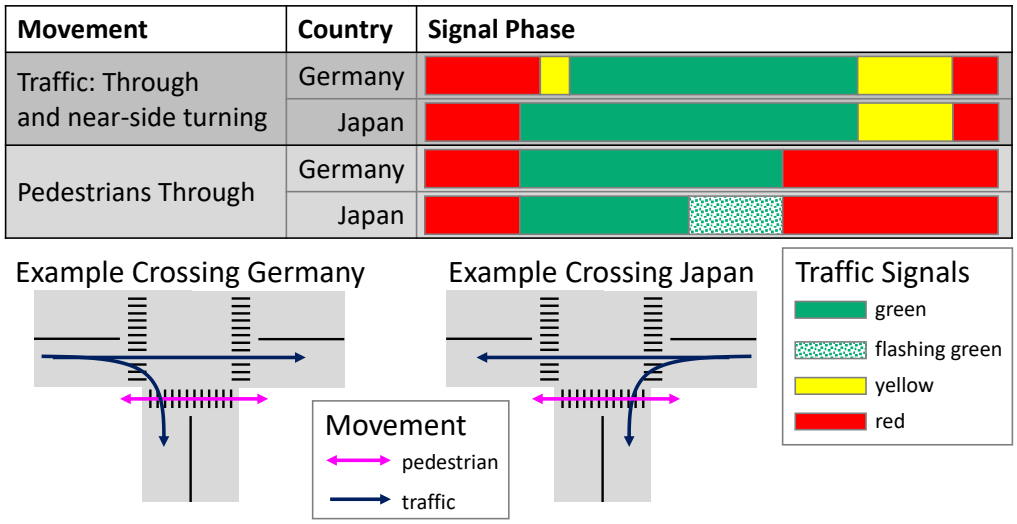


Figure 3.2: Simplified signal phases in Japan and Germany (cf. [WFF19; NIO19]). Reprinted with permission from [Hel+21] © 2021 IEEE.

There are, however, studies comparing Japanese and French pedestrians directly[24]. In those, differences between pedestrians initiating a crossing versus pedestrians following others into the crossing, were observed [PDS19b; PDS19a] Japanese pedestrians initialize a red-light crossing with 1.5%, less than French pedestrians with 55% [PDS19b; PDS19a]. Wait time and the duration of the light period influence the probability of illegal crossings, with a longer waiting time resulting in a higher likelihood of red-light crossings [Bro+13; PDS19a]. When crossing illegally, Japanese pedestrians initiate the crossing with a smaller time difference to the light change than French pedestrians [PDS19a] and only after the light for the vehicles turned to yellow [PDS19b]. This highlights the influence of culture on crossing behavior and indicates a higher level of compliance among Japanese pedestrians. A visualization of the relevant variables influencing waiting times is provided in Figure 3.3.

Findings concerning the departure time are not consistent, showing that French pedestrians are more likely to cross sooner legally after the light turns green [Pel+17], as well as the opposite case [PDS19a].

As the presence of other waiting pedestrians exerts social pressure on the rule compliance of others, the probability of a pedestrian crossing at the red light decreases as the number of waiting pedestrians increases [Ros09; Pel+17]. This effect is larger in Japan, reducing up to 70% of red light violations than in France (-37%) [Pel+17]. Other crossing initiators, however, may encourage waiting pedestrians to follow them against red. The probability of following depends on many factors, like proximity and gender [FKK10], and is based on an amplification process called mimetism or mimetic process [Ros09; FKK10; Cam+20]. In one study, Japanese pedestrians were only half as likely to be influenced by other crossing pedestrians as their French counter-

[24] A strong similarity between northern French and German pedestrian behavior is assumed due to their proximity and shared EU norms in traffic regulations. Japan-France-comparing studies are consulted due to a lack of direct studies comparing compliance between Japan and Germany. There is a heterogeneity of traffic safety [ASY14] and compliance levels [Sue+13; ZRA20] between Japan and other Asian countries.

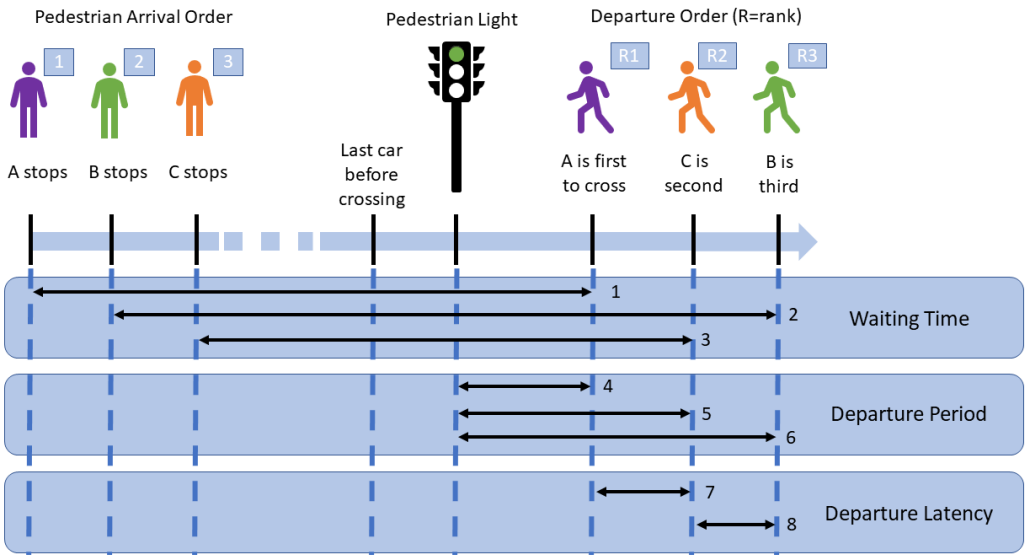


Figure 3.3: Visualization of the different culture-relevant variables. Waiting times (1-3) and departure periods (4-6) refer to pedestrians (A-C). (7-8) are the departure latencies of C (after the departure of A) and B (after the departure of C). (cf. [Pel+17]). Reprinted with permission from [Hel+21] © 2021 IEEE.

parts [Pel+17]. However, further analyses show that Japanese pedestrians incorporate both groups into their decision-making: pedestrians already crossing and pedestrians still waiting. On the other hand, French pedestrians tend to base their decisions purely on other already crossing pedestrians [PDS19b]. This reflects a comparatively more expansive mimetic process for Japanese that comprises more social environment cues than French pedestrians. Departure latencies do not differ between the countries [Pel+17]. There are no definitive results on the following behavior in Germany. Consecutive crossings with a pedestrian island change compliance rate in Germany. Rule violations at the first crossing (3%) are significantly less frequent than at the second crossing (39%) [Lan+16]. Such multi-stage crossings, however, are less frequently implemented in Japan.

Gap Acceptance. Pedestrians in Japan and France required a similar decision time before forming a crossing decision, but French pedestrians required a smaller gap (9 s) compared to Japanese pedestrians (16 s) to consider a crossing safe (accept it) in an observational study in an area with a speed limit of 50 km/h [Sue+13]. In Japan, female pedestrians require larger gaps (19 s) compared to male pedestrians (11 s) [Sue+13]. In Germany, different studies show that pedestrians require a time gap of 5 - 6 seconds to decide on a safe crossing, both in a real environment [SF09; BE14], as well as in a simulated environment [PNB17] with a speed limit of 50 km/h. These findings are consistent with the higher risk aversion in Japan and correspond to the gap acceptance. The frequency and duration of uncertainty behavior (freezing, abandoning, accelerating) during the crossing at red is also more pronounced in Japan compared to France (10%, $1\dot{5}$ s vs. 5%, $1\dot{2}2$ s) [Jay+20]. The difference is particularly pronounced when Japanese walk alone but is also visible among followers. In contrast, the hesitation tendency is purely observed among initiators

in France [Jay+20]. It was suggested that Japanese sometimes tend to cross inadvertently at red by solely relying on social cues, even if provided by rule violators. Japanese perceive transportation in general, including driving by car or bike, as less safe than Western countries, such as the USA [KR91].

Velocity and Gait. Japanese pedestrians move with an average velocity of 1.5 - 2.4 m/s during the crossing [ZW11; ZNW19] starting with smaller velocities (1 - 2 m/s) in the first half and ending with larger velocities (2.5 - 3.5 m/s) in the second half of the signal time [IA17]. This is most likely an effect of the onset of the pedestrian flashing green signal (PFG), which is specific for Japan [IA17]. In addition, pedestrians were found to accelerate in areas that conflicted with the traffic. When crossing at signalized crossroads, pedestrians show a higher velocity on the lane where the flowing traffic could turn into, regardless of which direction the pedestrians approached the crossing [IA17]. For unsignalized crossings, it was found that the implementation of refuge islands decreases the pedestrian velocity from 2.4 m/s to 1.5 m/s [ZNW19]. This is most likely an effect of a reduced perceived crossing risk [ZNW19]. Even though there are no PFGs in Germany, the light may switch to red during pedestrian crossing and could induce a change in velocity. Observational data from Germany, however, does not show any such change, as pedestrians show a very high continuity in their velocity (95.59%) [BE14]. Even more, most of the pedestrians are either walking (appr. 75%, 1.21 - 1.54 m/s) or fast walking (appr. 21%, 1.57 - 1.72 m/s) [BE14]. Although pedestrians do run on some occasions, it is not frequently observed.

3.1.4 Experimental Paradigms

Several experimental paradigms can be employed when analyzing pedestrians. While observational studies usually result in the most realistic behavior [Oxl+97], they involve many confounding events (e.g., other pedestrians and traffic density variation), making data extraction difficult. Experimental studies, on the other hand, enable the controlled variation of parameters and, thus, hypothesis-driven experimentation. Experimental studies of pedestrians in real environments are difficult due to the inherent risk of accidents for participants, confounding variables regarding other traffic participants, and high experimental overhead for planning, instrumentation of environments, and transporting researchers and participants to the real environment. Experimental studies in virtual environments allow easier data processing, higher participant safety, and excellent comparability between groups. However, it can be assumed that the validity of data captured in real environments is usually higher.

Different techniques can be employed when considering experimental studies using virtual environments. In multi-screen environments (e.g., [SGS08]), multiple displays show the street environment, and the participant can provide information via game controllers or indication of crossing intention. In cave systems (e.g., [LC07]), multiple displays or projections surround the participant, allowing for a 360-degree view. In addition, participants can move inside this environment, although using treadmills is more common for longer distances [SB20]. Head-mounted displays (HMDs, e.g. [FKK20]) can result in a higher immersion and better perception of crossing

affordances [Pal+21]. For further information on VR simulator studies of pedestrian behavior, we refer to respective review articles, such as [SB20]. However, regardless of the VR technology utilized in past studies, the space that a participant could physically enter was highly restricted. This restriction stretched from a few steps [Fel+18; Cle+19; KHW19] up to a single lane crossing [Jia+16].

The scene complexity [Oxl+97] and answer modality [LC07] influence the absolute decision criterion of pedestrian crossing behavior. In studies, participants indicate the crossing intention by shouting [SGS08], pressing a button [LC07], taking a step forward [SGS08; HH10], controlling an avatar with a joystick [CFG12], and crossing the street [Oxl+97; Jia+16]. The absolute gap (e.g., time-to-arrival) which was still considered as "safe" increases with a more realistic and complex environment [Oxl+97]. Thus, although results from small and simple environments or simple answer modalities can highlight important differences between participant groups, the absolute values cannot be directly transferred to real environments. Large virtual spaces with head-mounted displays allowing free movement in all directions and across the whole street should generally provide the most realism and, thus, most comparable absolute values.

3.2 Capturing Pedestrians in Real Environments

Although children are taught in primary school in Germany to extend their arm forward to express their crossing decision, this behavior is not common among adults. Thus, the primary indication of a crossing intention is expressed more subtly through the trajectory (like waiting at the curb) and body movements (like looking left and right for a gap in traffic). While this behavior is visible in everyday encounters, the articulated movements of pedestrians are almost neglected in research, except in the work of Kalantarov et al. [KRO18]. Due to the use of pose prediction in autonomous vehicles [ZB23], it is important to investigate the movements indicating pedestrian intention in more detail. In this section, an experimental study capturing and analyzing pedestrian motion data in real crossing scenarios is presented. The experiment focuses on two very similar actions: (a) crossing a street and (b) looking over the shoulder for a bus. The dataset provides novel insights into the motion of pedestrians in these two very similar actions and can be utilized in the development of elaborate motion models.

3.2.1 Method

The motion capture dataset was generated during a controlled experiment in a real street crossing scenario. The study was evaluated and approved by the Ethical Review Board of the Department of Computer Science (No. 18-11-2).

Participants. A total of 20 participants (10 female) took part in the motion capture experiment. All participants were recruited from the student body of the Saarland University and different schools in the area. The participants were between the ages of 18 and 34 (22.10 ± 3.48 years), of below average height (1.73 ± 0.09 m) and weight (66.15 ± 8.77 kg) [Sta23]. No participant was physically or mentally impaired, and all participants had normal or corrected to normal vision.

Material. The motion was captured using the Perception Neuron 32 suit[25], using 19 inertial measurement units (IMUs) for full-body capturing (excluding hands). To correct the global drift, the global trajectory was tracked using a Microsoft HoloLens 2 Development Edition[26]. Additional reference videos were recorded, displaying the motion as well as the surrounding street environment. In a post-processing step, the tracking and motion data were aligned. A geometric footstep detection was applied to identify individual footsteps. The tracking device's measurement errors (e.g., missing data) were corrected by applying foot-sliding avoidance (applying the inverse motion of a foot on the ground to the pelvis). The resulting motion was globally aligned to the street. The resulting motion clip was visually compared against the reference video.

Procedure. The experiment was conducted on a one-lane, two-directional main road on the campus (30 km/h limit). A zebra crossing enabled safe crossing. The traffic density varied depending on the time of day. However, vehicles usually exceed the speed limit and do not always yield to approaching pedestrians. All participants performed three walking tasks. The independent

[25]https://www.noitom.com/perception-neuron-series
[26]https://learn.microsoft.com/de-de/hololens/hololens2-hardware

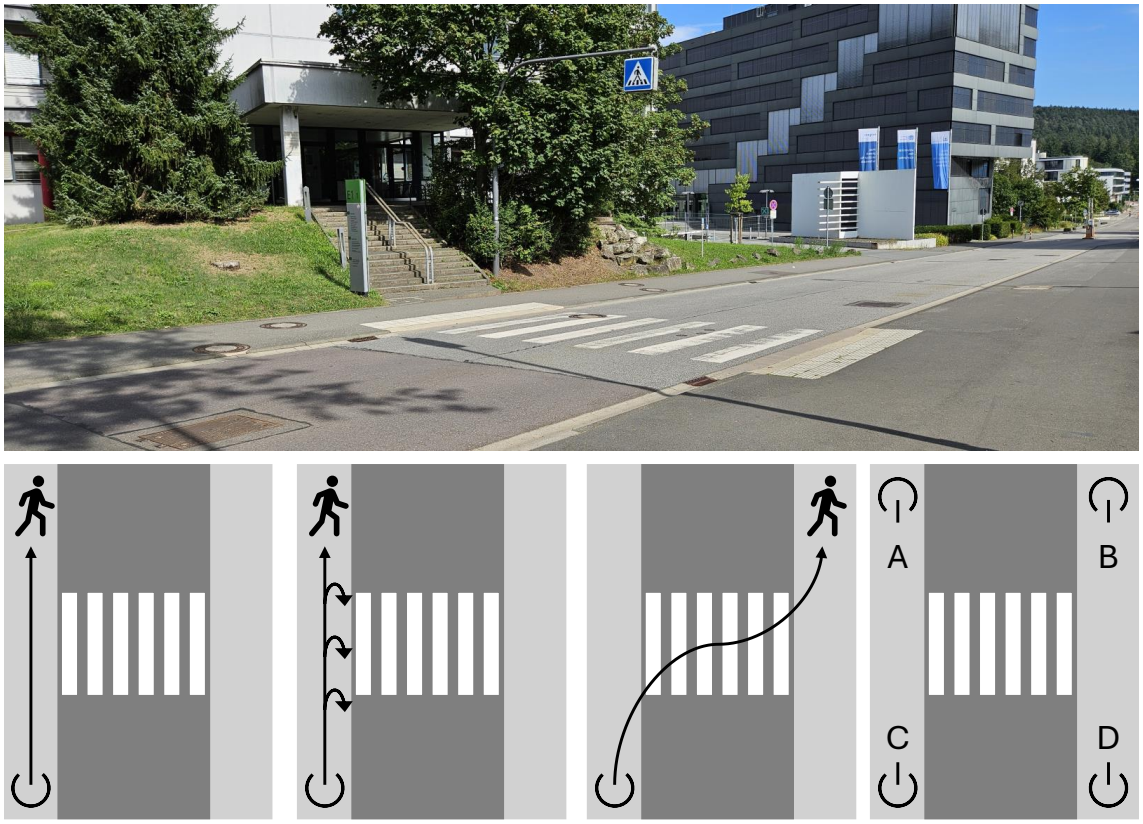


Figure 3.4: Experiment setup visualizing all three conditions and a photo of the experimental setting. All four starting positions are marked with a partially opened circle. (cf. [Spr+19a])

variables were walking task and walking direction. In a focused analysis of the captured motion, the dependent variables were body rotation and velocity. The three walking tasks were:

- WALKING: participants were instructed to walk alongside the street without further instructions as a baseline task.
- BUS: participants were tasked to walk alongside the street. As a cover story, they were told to catch a bus and were instructed to look for the bus, which would approach from behind. They were further instructed to ignore any approaching bus.
- CROSSING: participants were instructed to cross the street. As this task consisted of two fundamentally different phases, the results before the crossing event (BEFORE CROSSING) were analyzed separately from the results during the crossing event (DURING CROSSING). The first step on the street was considered as the criterion to separate both phases.

Two walking directions with respect to the street were distinguished:

- walking (and crossing) with the direction of the near lane
- walking (and crossing) against the direction of the near lane

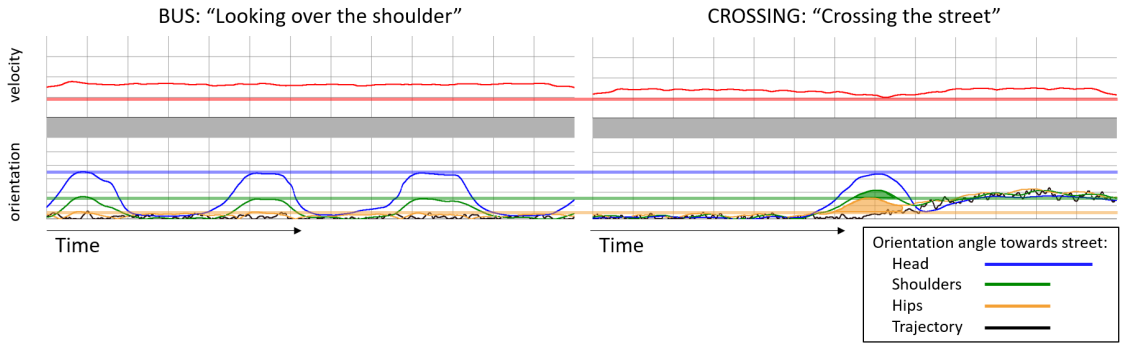


Figure 3.5: Exemplary results for a single participant. The velocity over time is smoothed in order to remove high-frequency noise, and the minimal velocity is marked. The absolute orientation of the head, shoulders, hips, and trajectory with respect to the street are presented. The maximum rotations in the BUS condition are marked, and the larger rotations of hips and shoulders before the crossing event are highlighted. Reprinted with permission from [Spr+19a].

In order to balance the conditions, each participant started from each side of the street (north, south) and from each side along the street (east, west). One examiner was placed at the start position, and one examiner was placed at the target position. Participants were instructed to walk towards the target examiner and return to the start examiner, continuing the same task (walking, bus, crossing). Figure 3.4 visualizes the setup. Hence, for each task and walking direction, a total of four trials were recorded and averaged.

Statistical Analysis. From the whole body motion capture data, the maximal rotation of the trajectory (global walking direction), head, shoulders, and hips with respect to the street (`body area`), as well as the minimal velocity, was analyzed.

All statistics were computed using the software package IBM SPSS, Version 25 [IBM17]. For pairwise comparisons, all variables were tested for normal distribution using the Shapiro-Wilk test [SW65]. The assumption of normal distribution was violated in some variables. Consequently, non-parametric tests were used. The same holds for the assumption of the equality of variances, which was tested and corrected for using Levene's test [Lev60] when required. In this work, effect sizes are not reported, as they can be computed trivially [Ras+14, p. 129]. Repeated measurement ANOVA were performed for multiple comparisons. Sphericity was violated in several comparisons, and Greenhouse-Geisser correction was applied. In order to enable the reader to follow the factorial design, uncorrected degrees of freedom are reported for F-values but p-values are reported using Greenhouse-Geisser correction.

3.2.2 Results

Our data suggest that pedestrians rotated their shoulders, hips, and trajectories more towards the street before crossing compared to looking for the bus. In addition, they slowed down before crossing. The head rotation, however, was not a distinctive feature for the differentiation of both conditions. Exemplary results for a single participant can be found in Figure 3.5. A total of 80

Table 3.1: Number of events and average duration of events for each experimental condition. Before (B) and during (D) crossing are abbreviated. Reprinted with permission from [Spr+19a].

Condition	Events	Average Duration
WALKING	80	12.85 s
B. CROSSING	80	7.45 s
D. CROSSING	80	4.36 s
BUS	258	3.85 s

crossing events can be compared against the same amount of walking events and a total of 258 bus events. Further information about the generated dataset can be found in Table 3.1. In the following, we present the statistics in more detail.

A repeated measurement ANOVA was performed with the factors `walking direction` (2) x `task` (4) x `body area` (4) and the dependent variable `maximal rotation` revealed a significant main effect of `task`, $F(3, 57) = 230.359; p < 0.001$ and of `body area`, $F(3, 57) = 405.499$; $p < 0.001$. In addition, the interaction between `task` and `body area` was significant, $F(9, 171) = 201.002; p < 0.001$. Post-hoc analyses using estimated marginal means revealed a significant difference in the maximal shoulder rotation, hips, and trajectory between BUS and BEFORE CROSSING. The difference in head rotation, however, was not significant.

A repeated measurement ANOVA with the factors `walking direction` (2) x `task` (4) and the dependent variable `minimal walking velocity` revealed a significant main effect of `task`, $F(3, 57) = 25.367; p < 0.001$. Post-hoc analyses using estimated marginal means revealed that the minimum velocity in BEFORE CROSSING is the lowest compared to all other tasks.

3.2.3 Discussion

The results of our evaluation suggest that pedestrian intention cannot be solely determined by head movement or past trajectory. Comparing an uncritical behavior (looking for the bus) with a critical behavior (crossing the street) suggests that the future walking direction can be more distinctively recognized by observing the hip and shoulder rotations. Qualitative analysis of the captured data revealed, that in crossing trials a larger rotation of these body parts can be observed before an actual change of trajectory. Hence, the exemplary actions could most likely not be accurately distinguished by a system that relies only on the trajectory and head orientation before the actual road-crossing event. Trajectory velocity, however, still seems to be a valid differentiation criterion. As hip and shoulder rotation precedes the trajectory change, considering these features during path prediction may improve the predictive performance.

As the traffic was not controlled and unspecific instructions were given by design, the recorded motion contains a lot of variety. Some of the participants were crossing the street regardless of the approaching traffic. Others seemed to rely on additional acoustic cues, as they did not turn in cases where it was very silent and no car could be heard in the larger area. In some cases, participants were forced to stop to avoid a collision, although they had the right of way.

3.2.4 Challenges of Capturing in Real Environment

The major challenge of capturing pedestrians in real environments is the highly dynamic and uncontrolled environment. While conducting the experiment, any timeslots falling into the lunch break and to the end of lectures were excluded in order to avoid high-density pedestrian traffic. In addition, Friday afternoon sessions were excluded to avoid low-density vehicular traffic. Even so, traffic density differed between trials and between pedestrians.

In spring 2020, a secondary study extending our experiments to include crossroads was started. Due to the COVID-19 pandemic and the resulting quarantine measures, the experiment had to stop prematurely after only a few participants were recorded. After the initial hard lockdown, campus traffic significantly changed; thus, the experiments could not be continued on campus until late summer 2022. At this point, the new paradigm of experimentation outlined in the next section was already established.

3.3 Capturing Pedestrians in Virtual Environments

The next generation of intelligent vehicles will require robust and safe interaction with pedestrians. Long-horizon predictions of pedestrian intention and trajectory are required to allow an early adjustment of the ego-vehicle movement and to prevent emergency braking as much as possible. Solutions to perceive and interact with pedestrians need to be developed and tested in a safe environment without endangering real pedestrians. In addition, these solutions should be able to operate in different countries. For example, solutions developed in Germany should be fully operational in Japan and vice-versa. We are not aware of any direct experimental comparison of pedestrian behavior between these two countries or any other in which pedestrians of both countries interact in exactly the same environment. As mentioned before (Section 3.1.3), past studies conducted in each country separately suggest differences in rule compliance [Lan+16; Pel+17], uncertainty [Jay+20; Ins24], gap acceptance [Sue+13; Pet14], or self-selected velocity for crossing [BE14; IA17]. As such, assumptions made for pedestrian intention and path prediction can be violated, or training data might be mismatched in one of the countries if these differences are not properly considered.

Conducting pedestrian experiments in virtual environments enables a direct comparison of cultural groups while simultaneously enabling a very accurate recording of participants' state, posture, and gaze. There have been multiple virtual reality (VR) studies using multi-screen environments [SGS08], cave-systems [LC07] and head-mounted VR scenes [FKK20]. However, existing simulators do not offer enough walkable space to allow participants to move freely, particularly when considering the reeving behavior of turning into the traffic. However, improvements in head-mounted VR systems have enabled larger experimental areas and are easily transported between different experimental sites.

In this section, the results of an experimental cross-cultural study analyzing the pedestrian behavior of 120 participants in Germany and Japan are presented. For this purpose, a new simulation environment was developed in which participants can move freely at a self-selected route with self-selected velocity and style of locomotion. Figure 3.6 shows a conceptual overview of the experimental environment. An untethered, head-mounted VR display with external tracking was used for traffic visualization. Participants were asked to cross a virtual street in different scenarios based on our research hypotheses, which were defined after consolidating existing research conducted in both countries. The scenarios included crossing with and without a zebra crossing and crossing with or without a group of other pedestrians.

We could identify new insights for the gap selection, choice of velocity, following behavior, and route choice of pedestrians and present evidence for previously unknown cultural differences. Our findings are already relevant to the cultural adaptation of advanced driver assistance systems (ADAS) that interact with pedestrians, such as path-prediction algorithms. In addition, the captured data can be directly utilized to generate and validate digital test environments for self-driving cars.

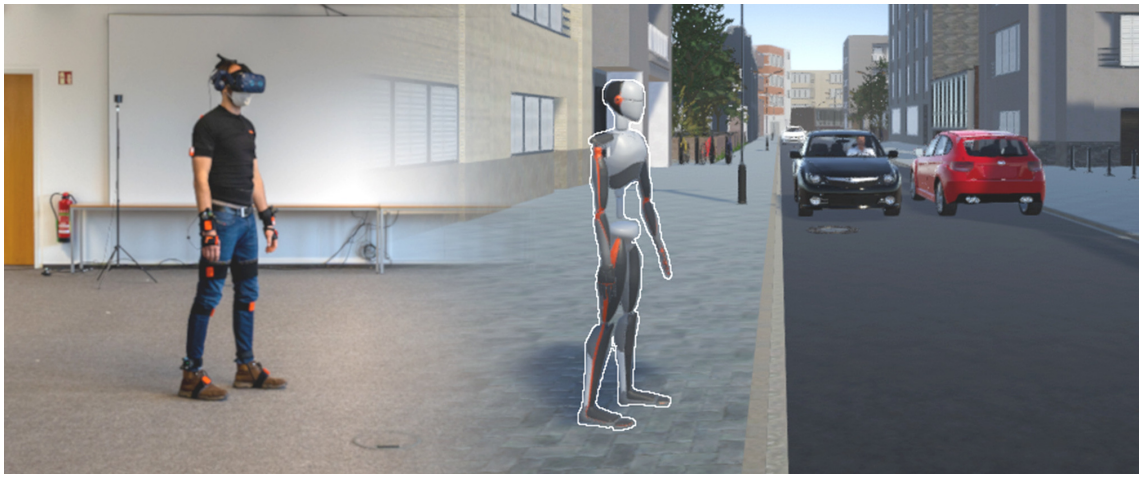


Figure 3.6: Exemplary visualization of the experimental setup comparing the real environment (left) and the virtual environment (right). Reprinted with permissions from [Spr+23] © 2023 IEEE.

3.3.1 Research Hypotheses

We investigated behavioral aspects according to the following new research hypotheses in our VR user studies in Japan and Germany.

For 'direct street-crossing' scenarios without any crossing facility:

- Japanese participants are expected to cross the street with higher velocity (*H1.a.*) [BE14; IA17],
- while requiring longer gaps to directly cross the street (*H1.b*) [Sue+13; Pet14].

When crossing together with other virtual pedestrians ('group crossings'), it is expected that based on a difference in the construal of self [Leb76; MK91]:

- the virtual pedestrians influence the gap choice of participants (*H2.a.*)
- the number of virtual pedestrians influences the behavior of Japanese more than of German participants (*H2.b.*)

For zebra-crossing scenarios ('zebra crossings'):

- Japanese participants are expected to have higher rule compliance [Lan+16; Pel+17] and thus utilize the zebra crossings more often (*H3.a.*)
- Japanese are expected to wait longer in front of the zebra crossing before crossing (*H3.b.*) due to a higher uncertainty [Hof01].

3.3.2 User Studies in Japan and Germany

Participants. In each country, 60 participants between the ages of 20 and 50 were recruited. The experiments took place in Saarbrücken, Germany, and Tokyo, Japan. Participants had to be within

Table 3.2: Descriptive statistic of all participants. Reprinted with permission from [Spr+23] © 2023 IEEE.

	GER	JPN
Participants	60	60
Female	30	30
Age ($years$)*	25.42 (5.93)[a]	35.70 (8.68)[a]
Height (cm)*	176.09 (8.66)[a]	168.83 (8.90)[a]
BMI (kg/m^2)*	23.02 (3.77)[a]	21.04 (2.20)[a]
Driving experience ($years$)	6.80 (6.10)[a]	12.03 (10.34)[a]

*Significant differences with a Bonferroni-corrected level of 0.00625
[a]mean values, standard deviations in brackets

a normal range of BMI (body mass index), without any known physical or mental disabilities, and normal or corrected to normal vision. The study was reviewed and approved by each country's local ethical review boards in advance (registration numbers in Japan: H2022-1166-B; and in Germany: 21-08-06).

The descriptive statistics of all participants can be found in Table 3.2. As expected, Japanese participants were smaller in height and weight (BMI) than German participants. Due to differences in recruiting, the German participant group was significantly younger than the Japanese age group. While participants were recruited from the local university in Germany, a recruiting company selected participants from the general population of the Kantō area. However, past research does not suggest an influence on the analyzed behavior within this age group.

Experimental Procedure. After preparation, participants started with a short initial familiarization and training period inside the virtual reality environment, which took 11 min on average. During familiarization, a collision with a vehicle was intentionally performed, satisfying participants' curiosity. Afterwards, participants showed visible relaxation and higher immersion. After familiarization, participants performed three training trials. The familiarization and training trials were not used for the evaluation. During the actual experiment, three different types of crossing scenarios were performed for the above-mentioned hypotheses. In 'direct crossing' (15 trials), participants had to cross the street alone. In 'zebra crossing' (15 trials), there was a zebra crossing 6.25 m away from the starting position, which participants were not required but allowed to use. In 'group crossings' (30 trials), there were additional virtual pedestrians crossing alongside the participant. Figure 3.8 shows the experimental setup for all different scenarios, and an example frame from an ego-perspective of a pedestrian can be found in Figure 3.7.

The 'direct' and 'zebra crossing' trials were grouped together in one block and shown separately from the 'group crossings'. The order of blocks was balanced over all participants. After every 20 trials, participants were offered a break. Due to Japan's high temperature and humidity, the room was air-conditioned, and breaks were mandatory. On average, participants required 11.63 minutes for 20 trials.

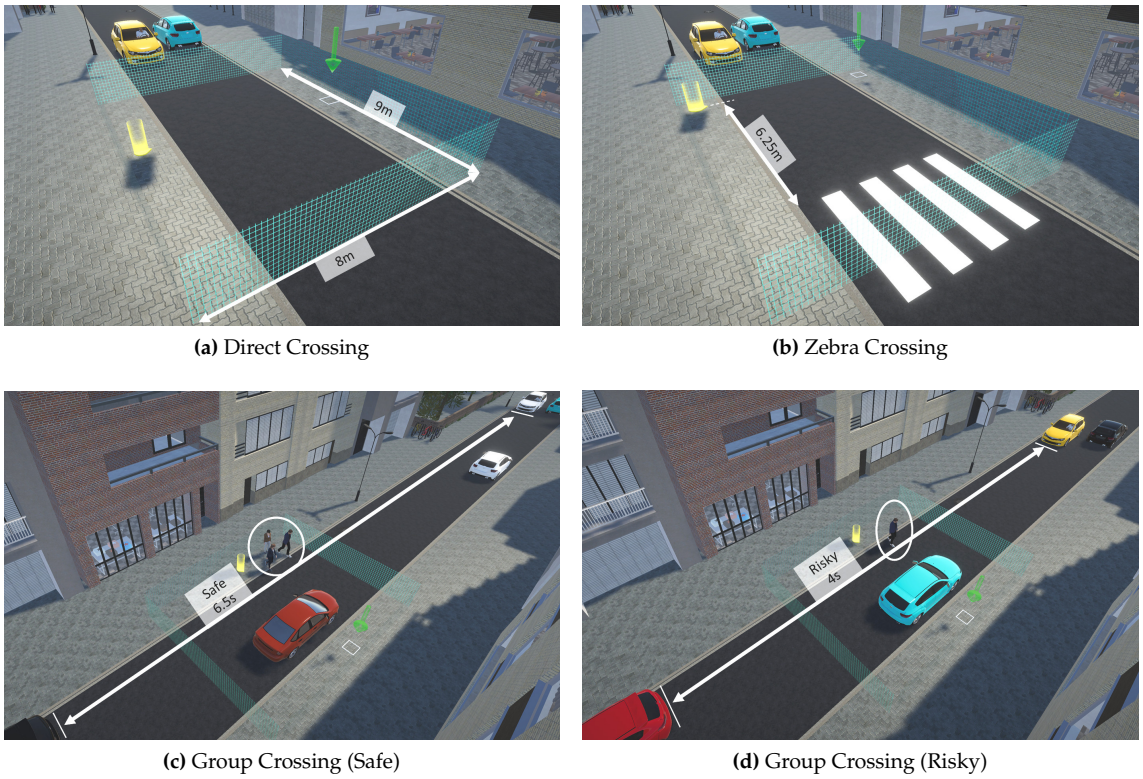
Figure 3.7: View from the perspective of a participant during the experiment. Reprinted with permission from [Spr+23] © 2023 IEEE.

3.3.3 Technical Set-Up

We developed a simulation environment to capture pedestrian behavior and movement. This virtual 3D environment was designed to represent a street scene in both countries, requiring only minor cultural adaptations. The proposed system allows the participants to walk and run freely in a 9 x 8 m area with a head-mounted VR headset and an IMU-based motion-capturing system.

Virtual Environment. Inside of the Unity game engine, a virtual street environment was constructed. It displays a two-lane road with opposing traffic. Each lane has a width of 2.25 m, and the street a total width of 4.8 m (including the gutter), a curb of 5 cm height and 10 cm width, and a 4.5 m wide sidewalk. The lanes are not separated by a center line. Participants are able to walk within 1.5 m of the sidewalk until a visual wall appears and indicates the bounds of the physical experimental space. The virtual street consists of a straight segment of 200 m, followed by a curve. The participant space is positioned in the center of the straight segment, allowing for a visual distance of up to 140 m in both directions. Additional houses prevent participants from perceiving the horizon in all directions. Several virtual small-sized sedans of different colors are displayed, driving along the predefined curve of the street. Cars are spawned outside of the visual space of the participants, are driving at a constant 30 km/h, and a virtual driver is sitting inside each vehicle in a driving position. Figure 3.8 shows a visualization of the environment.

In some trials, unsignalized zebra crossings with a width of three meters are displayed, and the start is moved to the corner of the walkable space to a distance of 6.25 m of the zebra crossing (see Figure 3.8b). When participants are within the vicinity of the zebra crossing, vehicles stop at the zebra crossing until the participant leaves the vicinity of the crossing again. Outside of the zebra crossing, vehicles do not stop for pedestrians, and in case of a collision, the trial is stopped, and an according message is displayed to the participant. Some trials also contained other virtual



(a) Direct Crossing

(b) Zebra Crossing

(c) Group Crossing (Safe)

(d) Group Crossing (Risky)

Figure 3.8: Experimental setup showing all types of trials. Start and goal positions were mirrored in every other trial to prevent street crossings outside the experiment. The example images here display right-handed (German) traffic. In all trials, the start position was in the middle of the walkable space, except for zebra crossings (b), where the start and goal were shifted to one side of the space. In group crossing scenarios ((c) and (d)), different combinations of pedestrians were shown. Pedestrian geometry models were randomized. Reprinted with permission from [Spr+23] © 2023 IEEE.

pedestrians in the scene. Two male and two female characters were created for each cultural environment and dressed in the same clothing. Virtual pedestrians crossed the street either in a 'safe gap' (6.5 s; Figure 3.8c) or a 'risky gap' (4 s; Figure 3.8d).

The experiment was separated into two blocks, with the first block containing 15 trials of direct crossing and 15 trials of zebra crossing. The second block contained only direct crossings, with virtual pedestrians crossing with the participant. Trials within each block were randomized, and the order of blocks was balanced according to the gender of the participants. After every 20 trials, participants were offered a break. As the experiment was conducted in Germany in winter and health regulations required frequent venting of the room between participants, most participants did not require the break. In Japan, the experiment was conducted in the summertime with high temperatures, high humidity, and restricted air conditioning to conserve power. Hence, every break was utilized, and participants were offered water.

Cultural Adaptations. Minor cultural adaptations of the traffic scenes were required to conduct a pedestrian study in different countries. Besides mirroring the street from right- (Germany) to left-handed (Japan) traffic, the UI elements were translated, and the body size and facial features

of the virtual pedestrians were adjusted. No participant complained about the authenticity of the environment or visualization of characters.

Hardware. The Vive Pro Eye[27] head-mounted VR goggles were utilized using the wireless adapter to allow participants to move uninhibited. Since the Vive wireless adapter was not certified in Japan at the time of the experiment, permission to use the device was obtained from the Ministry of Internal Affairs and Communications (Registration number: 01-20220524-03-997372). Using four SteamVR Basestations, a lighthouse environment of 9 x 8 m was created. The IMU-based motion-capturing solution MVN Awinda by Xsens using 17 IMU sensors was utilized to track the kinematic movements of all participants. The sensors were strapped to the body of participants with the provided straps and a shirt on top of their usual clothing. Additional Vive trackers were used for better spatial alignment [SHK22].

Dataset Replay. The recorded raw data contains only minimal information required to replay the whole experiment exactly, including state information, duration information, positional information, and motion capture synchronization for every frame. During replay, more computationally expensive and acausal operations can be performed, like image extraction or the identification of false starts. The only difference between the actual experiment and the replay is the stochastic image noise inherent to the render pipeline. The statistical evaluation presented in the results is performed on data, which was computed in the replay functionality. In addition, the replay functionality allows subsequent data extraction and evaluations that are already partially published [Zha+24a; Zha+24b].

3.3.4 Results

Non-parametric tests were used for statistical comparisons when the assumption of normal distribution was violated. Significance levels were adjusted using the Bonferroni correction, if required. The statistical evaluation was computed using R [R C22] version 4.2.1.

Direct Street Crossing. From the recorded data of all 'direct crossing' trials, the velocity was computed using the differences of position in a sliding window of one second. While on the pavement of the street, Japanese participants moved with a mean velocity of 1.51 m/s (SD = 0.18), while German participants moved with a velocity of 1.43 m/s (SD = 0.19). This difference was significant (t(117.92) = -2.444; p = .008), thus confirming our hypothesis of Japanese participants using a faster pace when crossing the street (*H1.a*).

As to our Hypothesis *H1.b*, the gap in traffic was computed as the largest time gap directly between two cars (including the opposing lanes). The gaps were computed over all 'direct crossing' trials. In this study, German participants accepted a mean gap of 5.94 s (SD = 0.43) and Japanese participants a mean gap of 6.11 s (SD = 0.44). Although the numerical difference between the average gap is only small (0.17 s), the difference was significant (t(117.95) = -2.152; p = .017). Thus, it confirms our hypothesis that Japanese participants require larger gaps in traffic to cross.

[27] https://www.vive.com/sea/product/vive-pro-eye

Before crossing, Japanese participants waited significantly longer (Mdn = 14.09 s) than German participants (Mdn = 10.23 s) to select a suitable gap ($U(N = 60) = 816$, $p < .001$) and rejected larger gaps (4.8 s, SD = 0.85) than German participants (4.07 s, SD = 0.81; $t(117.65) = -4.44$, $p < .001$). In addition, 23 Japanese participants were not able to complete crossings in time (60 s), in contrast to only one German participant in this respect.

These results suggest that Japanese pedestrians may have a higher desire for safety than German pedestrians, as they waited longer to accept a larger gap and then walked faster to cross the street. In addition, during the whole experiment, there were 119 aborted crossing attempts of the Japanese participants compared to 56 aborted crossing attempts of the German participants, which indicates an additional higher uncertainty before crossing.

Following Behavior. To test the influence of other pedestrians on the gap choice, direct crossing trials without virtual pedestrians (agents) were contrasted with trials in which these agents took either risky or safe gaps. The experiments revealed that the general behavior of all participants concerning accepted gaps is significantly influenced by the presence of virtual pedestrians (ANOVA $X^2(2) = 18.43$, $p < .001$), which confirms our Hypothesis *H2.a*. Pairwise comparisons revealed that the presence of an agent that safely crosses the street resulted in significantly larger accepted gaps (Mdn = 6.50 s) by the participant than in the presence of a risky agent (Mdn = 6.12 s, $p < .001$). Furthermore, a safe agent led the participants to more conservative gap choices (Mdn = 6.50 s) compared to trials in which participants crossed alone (Mdn = 6.10 s, $p < .001$), whereas a risky agent did not evoke riskier gap choices by the participants compared to crossing solo ($p = 1.00$).

The scenes in which several other agents crossed the street were selected to test the influence of the number of crossing pedestrians on the gap choice. The accepted gaps in the conditions of majority and minority, with either one or two agents crossing, and safety, in with agents crossing on a short or longer gap, were compared. Consistent with the previous analyses, there was a significant effect of safety ($Q = 21.64$, $p < .001$), showing again that safely behaving agents led to larger accepted gaps than risky agents. However, there was neither an effect of the actual number of agents crossing ($Q = 0.72$, $p = .396$) nor of culture ($Q = 0.27$, $p = .603$). Importantly, there also was no significant interaction between the number of agents crossing and of culture ($Q = 0.18$, $p = .670$). When the virtual pedestrians utilized a larger gap, participants also utilized a larger one. However, virtual pedestrians using very short gaps had no significant effect on the gap selection of participants. The results do not support a cultural difference nor an influence of the actual number of virtual pedestrian, neither in Germany nor in Japan, which falsifies our Hypothesis *H2.b*.

Use of Zebra Crossings. In total, 16 Japanese (7 German) participants always utilized the zebra crossing, while 12 Japanese (7 German) participants never did. During the post-questionnaire, participants confirmed their behavior to be representative of their real zebra crossing utilization. There was no significant difference in this respect between German and Japanese participants on average, which is against our Hypothesis *H3.a.* However, when utilizing the zebra crossings in the scenes, Japanese participants waited significantly longer (4.55 s, SD = 1.06) than German participants (2.95 s, SD = 1.19) before stepping on the street. These results confirm our Hypothesis *H3.b*.

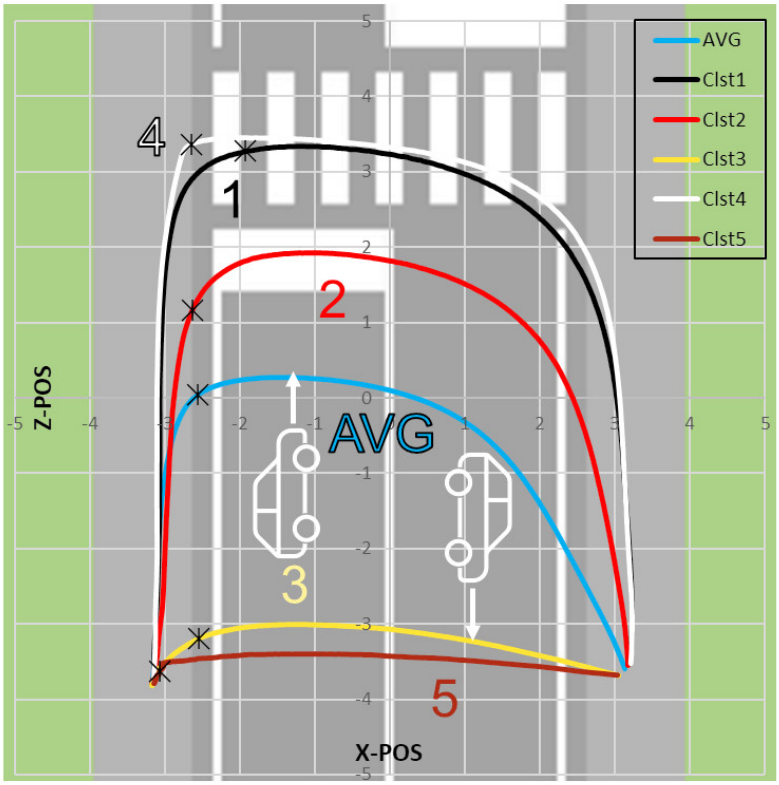


Figure 3.9: Reconstructed trajectories of the hierarchical cluster analysis. The German data was mirrored to the left-handed traffic for analysis. Stars [*] denote 50% of the crossing duration. Reprinted with permission from [Spr+23] © 2023 IEEE.

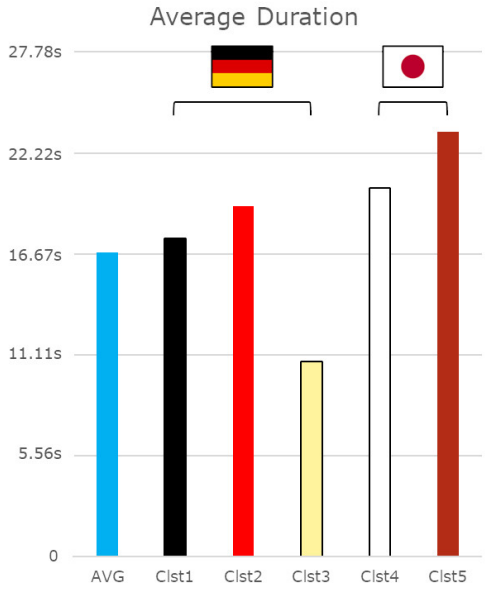


Figure 3.10: Average duration (time) of the trajectories of each cluster. Reprinted with permission from [Spr+23] © 2023 IEEE.

Table 3.3: Trials per cluster and country. Reprinted with permission from [Spr+23] © 2023 IEEE.

	Germany	Japan
Clst1	**136** trials	69 trials
Clst2	**210** trials	173 trials
Clst3	**350** trials	231 trials
Clst4	116 trials	**285** trials
Clst5	32 trials	**116** trials

Our experiments indicate that Japanese participants cross the street even further away from the zebra-crossing compared to German participants. This route choice behavior was analyzed in more detail by considering the full trajectories of all participants in 'zebra crossing' trials. Principle component analysis (PCA) was performed, and a hierarchical cluster analysis was computed on the first 21 components explaining 99% of variance. Clusters Clst1, Clst2, and Clst3 were primarily composed of trials of German participants, while clusters Clst4 and Clst5 contained mostly trials of Japanese participants (see Table 3.3). The statistical significance of the differences was confirmed (X^2(4, N = 1718) = 168.27, p $<$ .01). As seen in Figure 3.9, Japanese clusters show a clear distinction between either waiting and directly crossing (Clst5) or utilizing the zebra crossing (Clst4). German clusters (Clst1, Clst2, Clst3) show a clear influence of the zebra crossing, partially visible in a 'short-cutting' behavior or tendency to veer towards the zebra crossing.

Analyzing the average durations (see Figure 3.10) shows another important difference. Clst3 has a shorter duration than Clst1 and Clst2, hence there is a strong indication that a direct crossing was chosen as it enabled faster crossing. However, Clst5 has a longer duration than any other crossing, indicating the motivation to reduce the path length rather than the path duration. This temporal effect is highlighted in Figure 3.9 with a star denoting the half-time point of the crossing duration.

3.3.5 Discussion

Using the virtual reality setup, it is possible to capture and compare pedestrian behavior between participants in Germany and Japan.

Implications for Intelligent Vehicles. In line with our hypotheses, our experimental results suggest a higher desire for safety on direct crossings for Japanese pedestrians, indicated by a higher crossing velocity and by requiring larger gaps in traffic. As hypothesized, we found indications for a higher uncertainty in the Japanese group, as Japanese participants aborted a crossing attempt more frequently than German participants. In addition, Japanese participants waited longer in front of a zebra crossing before stepping onto the pavement. There are several implications of these cultural differences for the design and implementation of intelligent vehicles.

Indication of a vehicle's yielding (e.g., by reducing velocity, optical signals) should be performed earlier in Japan than in Germany to account for their higher gap requirement and uncertainty. At zebra crossings, this is even more important: Pedestrians should not wait at the zebra crossing until the approaching car stops, as this would inhibit a smooth flow of traffic. Autonomous vehicles

should obey traffic rules and yield to pedestrians, and they need to indicate their yielding to enable pedestrians to start crossing before coming to a complete stop.

Regarding 'group crossings', a clear influence of the presence of virtual people on the selection of gaps in traffic was shown. More specifically, a virtual pedestrian selecting a safe gap would lead to the selection of a safer gap by participants. However, we could not identify a cultural difference in this behavior. Consequently, ADAS should always consider all pedestrians together when predicting the intention of an individual pedestrian in a group.

Cultural differences in route choice behavior at zebra crossings were opposite to our expectations: while we did expect Japanese pedestrians to utilize the zebra crossing more often, a more diverse behavior was observed. German participants tended to cross the street on an arch, short-cutting the entrance and exit of the crossing. Japanese participants, however, either utilized the zebra crossing and waited before entering it or ignored the crossing completely and crossed the street on a direct route. This behavior was chosen, even if waiting for a suitable gap would take longer than taking the path with a longer distance utilizing the zebra crossing. Multiple Japanese participants confirmed their strategy of either using the zebra always or never in the post-questionnaire.

To the best of our knowledge, this behavior has not yet been reported in the literature, and it contradicts the often-stated hypothesis that Japanese people are more cautious. As this points towards a systematic difference in trajectory between Japanese and German pedestrians, it is highly important for pedestrian path prediction solutions for highly automated driving and must be considered when training and testing these systems.

Limitations. There are some limitations of our study: First, the influence of height difference and the influence of past accident experiences were not further investigated and could, potentially, influence the gap acceptance. Second, gap duration was uniformly sampled between 2.5 s and 8.5 s and between cars approaching from both or only a single lane. This could result in sequences of very short gaps or sequences of very long gaps.

Future Work. Large-scale VR experiments are a useful tool to investigate complex pedestrian behavior. The data captured can be utilized to develop pedestrian software agents for the simulation of test scenarios in a digital reality [Dah+19]. Specifically, when including the motion capture data, pedestrian agents can be built that behave as their natural counterparts and move their bodies accordingly, enabling a more realistic simulation for image- and posture-based prediction algorithms. Further analysis of the gaze, body postures, and movement dynamics is required to support more information in that regard. The presented experimental setup seems particularly useful for capturing and analyzing risky scenarios in the future (e.g., near misses) that cannot be created in real environments, as well as experiments with high-risk groups.

While we have captured the motions of all participants, the detailed analysis and modeling of this data remains open and was not performed during this dissertation due to the chronological order of the conducted work.

3.4 Towards Capturing Child Behavior and Motions

Children participate in urban traffic, and thus, the next generation of autonomous vehicles will require robust and safe interaction with them as well. Solutions to perceive and interact with children need to be developed and tested in a safe environment in which pre-crash simulations can be performed without potentially harming real children. Game-like environments (e.g. [Dos+17]) can provide the technological basis for simulation but require realistic agents to create valid training and testing scenarios for the autonomous vehicle and exhibit correct trajectories and motion cues.

To create these realistic agents, we require sufficient high-quality data of child behavior and movements in uncritical as well as critical situations. Pedestrian experiments in virtual reality (VR) environments can be utilized to capture the required state, posture, and gaze data of participants (e.g. [LC07; Jia+16; Cle+19; KHW19; SB20; FKK20; Pal+21; Spr+23]). However, there are only a few examples in which these experiments were conducted with children in the ages 5-18 (e.g. [SJR03; CFG12; SMS15; Wan+22]) in very simple environments. Particularly for younger children (below 13 years), the lack of experience and cognitive development influences their decision-making and crossing behavior [Wan+22] and therefore needs specific investigation.

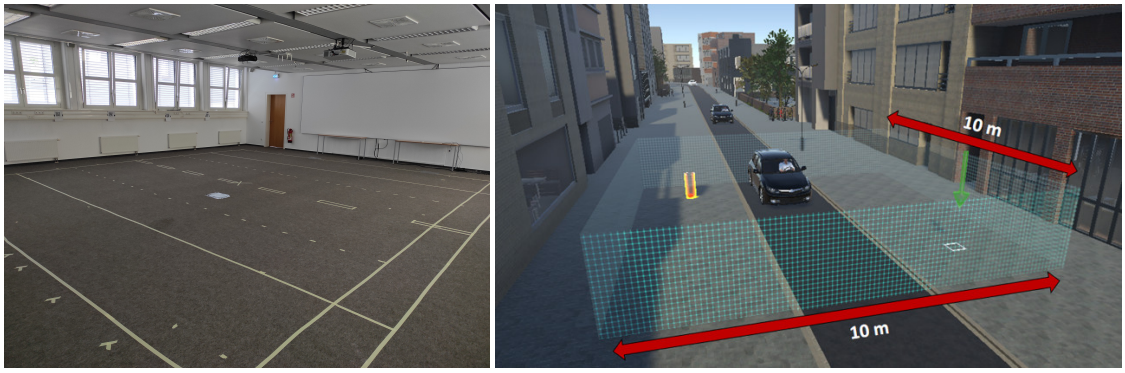
In this section, the results of a first feasibility study analyzing the potential of large-scale VR experiments to capture and evaluate the behavior and motions of children are presented.

3.4.1 Method

We extended our existing simulation environment [Spr+23] with new scenarios and adjusted it to run on an untethered, child-friendly, head-mounted VR display (Vive XR Elite[28]). Motions were captured using the Xsens Awinda motion capture suit. The environment allowed free movement on a 10 x 10 m area in eight different scenarios. Four environments were established: (i) without any crossing facilities (see Figure 3.11), (ii) with a zebra crossing, (iii) with a traffic light (see Figure 3.13), and (iv) with parked vehicles occluding the approaching traffic (see Figure 3.12). In addition, we constructed a one-way and a two-way road with opposing traffic for each type of environment. Children were provided realistic VR scenarios to encourage them to act as they would in real-life traffic scenarios. Additionally, the scenes were built with the feature to accurately reconstruct their street crossing behavior and extract the resulting behavior data. In total, the participants were asked to cross the road 21 times, and trials differed in their environment and traffic density.

We invited 12 participants between 5 and 13 years old, accompanied by their parents. While the total experiment lasted for about an hour, the children used the VR headset for 20 - 25 minutes. After 20 minutes, a break was mandatory to prevent exhaustion and eye fatigue. The children and their parents were allowed to pause or end the experiment at any point. The study design and procedure were approved by the DFKI ethics board.

[28] https://www.vive.com/de/product/vive-xr-elite



(a) Real Lab Environment **(b)** Virtual Street Environment

Figure 3.11: Real and virtual environment. The left figure displays the real lab environment containing markings on the floor to visualize the virtual street (displayed on the right) to the parents. Inside the virtual environment, the yellow cylinder displays the start and the green arrow the goal position.

(a) Occlusion Scene **(b)** Participant View

Figure 3.12: Scene and participant view for the occlusion trials, in which the participant had to enter the environment from behind the occlusion (between two cars).

(a) Zebra Scene **(b)** Traffic Light Scene

Figure 3.13: Exemplary visualizations of the zebra and traffic light environments. All environments within the experiment were displayed in both, one-lane and two-lane layouts.

3.4.2 Results

While we were successful in conducting the experiment with all school children, four out of five preschool children aborted the experiment prematurely. Two of them did not feel comfortable starting the experiment at all, while the other two children aborted the experiment after multiple trials. After excluding all trials with collisions, timeouts, and connection losses, a total of 183 valid trial recordings remained. Due to the small number of remaining trials and the heterogeny of participants, no statistical tests were performed, and all following comparisons were made on a purely numerical basis and might not be statistically relevant.

Collisions. Eight collisions occurred during the valid trials. Four participants had one collision each, and one participant had four collision events. Collisions occurred in different environments and on both, one- and two-lane roads.

Wait Time. Before crossing the road, participants missed on average 1.64 gaps (SD = 1.07), which would have been feasible to cross. The children waited on average 19.6 s (SD = 8.68) before entering the road surface. The differences between trial types (baseline and occlusion) and the number of lanes were marginal. In addition, a total of 10 crossing attempts were not completed within a 60 s time window.

Gap Acceptance. In non-controlled scenarios (baseline, occlusion), a gap was selected with a time-to-collision (TTC) of 7.67 s (SD = 2.62) when entering the road and a safety margin of 5.87 s (SD = 2.53) when exiting the pavement. In occlusion trials, gaps with longer TTCs (8.11 s, SD = 3.36) and safety margin (6.45 s, SD = 3.38) were selected than in baseline trials (TTC: 7.41 s, SD = 2.11; safety margin: 5.56 s, SD = 1.89). In two-lane roads, gaps with a slightly shorter TTC (7.47 s, SD = 2.54) and safety margin (5.64 s, SD = 2.01) compared to a one-lane road (TTC: 7.81 s, SD = 2.69; safety-margin: 6.06 s, SD = 2.89) were selected. All selected gaps are larger than the gaps observed on adult pedestrians in similar environments [Spr+23].

Zebra and Traffic Signal Utilization. Out of 52 trials within the zebra environment, the zebra was not utilized in only four trials, containing both one- and two-lane roads. Participants occupied the space in front of the zebra for 5.12 s (SD = 2.74) before entering the pavement, which is longer than adults in similar environments [Spr+23]. In addition, the children did not show short-cutting behavior but always crossed the road fully on the zebra, which again is dissimilar to adults [Spr+23]. Two participants did not utilize the signalized crossing in six out of 41 trials with traffic signals. In all trials in which the signal was utilized, participants waited until the pedestrian light turned green and did not directly violate the traffic light.

Velocity. In the baseline trials, the participants moved with an average velocity of 1.66 m/s (SD = 0.49) on the pavement, and thus considerably faster than in occlusion trials (1.18 m/s, SD = 0.48), in zebra trials (1.23 m/s, SD = 0.41), in traffic signal trials (1.40 m/s, SD = 0.49), and on the sidewalk (0.36 m/s, SD = 0.20). In addition, they moved faster on two-lane roads (1.62 m/s, SD = 0.51) than on one-lane roads (1.26 m/s, 0.51). The maximum velocity was 2.76 m/s, which is approaching a jogging velocity for the observed age range [EH77]. The velocity of the children on

the road was generally faster than adult pedestrians in comparable environments [Spr+23] even though the children had the shorter legs.

Route Choice. While the trials containing zebra crossings and traffic signals predominantly contained three distinct phases for almost all participants (walk to crossing facility, cross, approach target), the other trials were more complex. In many cases, participants did not directly cross the road but had false starts (21 times), waited on the street to let traffic pass (21 times), or waited on the center line to avoid collisions (one time). Particularly in the case of parked vehicles occluding the view, participants had to step on the pavement to see the approaching traffic.

Body Movements. While most participants remained in a rather neutral walking style at the beginning, the style did change over time for some children. Similar to adults [Spr+23], most children visibly relaxed after a few trials. Some increased the exaggeration of their arm movements, and most tried to press the traffic light button. Most children switched to a running gait at least once throughout the experiment. In the occlusion trials, some looked around the cars while others tried to peek through the windows of the parked vehicles.

3.4.3 Discussion

This study shows the general feasibility of conducting pedestrian VR studies with school children. While capturing motions with preschool children is generally practicable, conducting pedestrian VR studies in complex environments with them is challenging due to low attentional capacities, rapid fatigue, and sometimes anxiety and is thus considered infeasible.

While our sample size was too small to enable clear statistical evaluations, initial indications of their behavior are reported. Most children adhered to the traffic regulations and selected safer routes and longer gaps than adults, but they moved with a higher velocity. The general uncertainty was higher, especially in environments with restricted visibility.

The captured data enables the subsequent modeling of pedestrian agents and their animations to generate more realistic pre-crash simulations. Although the number of participants is insufficient for this task, nor for a more detailed evaluation, our results provide a perfect basis for implementing and configuring a more extensive study. Further analysis and modeling of the motion remain open tasks for the future.

Lastly, while our study is similar in design and tasks compared to [Spr+23], it does not have the same conditions. For more detailed comparisons between children and adults, adults would need to perform the same tasks in the same environments.

3.5 Chapter Discussion and Future Work

3.5.1 Discussion

In the first study in a real environment (Section 3.2), the body movements before entering the road differed significantly from not entering the road. Shoulder and especially hip rotations provide significant information on the crossing decision, before the pedestrian adjusts the trajectory, while the head orientation itself is not a distinct criterion. Thus, shoulder and hip rotations are a beneficial cue for autonomous vehicles and should be represented in simulated environments accordingly. While the captured behavior can be considered to be highly realistic, as it was captured in an actual street environment, the data quality itself is debatable. The positioning and alignment of captured data to a digital model of the street is complicated and needs manual post-correction. The annotation of other participants (e.g., other pedestrians and vehicles) cannot be easily controlled and needs also to be manually annotated from surrounding video captures. It is, therefore, not a cost-effective paradigm if other information additional to the actual body pose is required. In addition, the organizational overhead of outside capturing cannot be underestimated. Finding a suitable location for capturing, bringing experimenters, participants, and equipment to the location, and dealing with external influences like weather, street repairs, and illegally parked vehicles add complexity.

We conducted a large-scale virtual street crossing environment, using a head-mounted display combined with full-body motion capturing to conduct the experiments in a lab environment (Section 3.3). Besides the benefit of having fewer uncontrolled variables, an automatic recording of all traffic participants, and accurate participant positioning on the street, we could also conduct the same study within Germany and Japan and investigate the effect of culture on pedestrian behavior. Well-established effects were reproduced in each country separately and compared between both countries. As both participant groups crossed the same street, new effects with a significant influence of culture could be identified that had not been reported before. Of these, the selection of trajectories is particularly relevant for pedestrian agent simulation. The German participant group shows a shortcutting behavior, dynamically entering the road before reaching the zebra crossing. The Japanese participant group, however, displays a split decision, either utilizing the zebra crossing and waiting in front of it until the traffic stops or ignoring the zebra crossing and crossing directly. In both cases, autonomous vehicles must correctly identify the crossing intention and consider it in the navigation planning. The quality of the captured data is usable for motion modeling, trajectory prediction, crossing prediction, pedestrian steering, and behavior modeling. Different research groups utilize the data, and additional insights were recently published [Zha+24b].

A virtual reality environment is a safe environment without the possibility of collisions of pedestrians and vehicles. It is thus possible to conduct studies with pedestrians in high-risk groups in which experimental crossing experiments are not safe to be conducted in real environments. The

generalization to high-risk groups was evaluated in a feasibility study with children in Section 3.4. The results indicate different movement patterns compared to adult pedestrians. Besides requiring larger gaps and showing an increased need for safety, the children displayed a distinct waiting period before entering the road in front of the zebra crossing, which was more similar to the Japanese population than the German population.

3.5.2 Future Work

One of our core assumptions is that pedestrians in a virtual reality study show similar behavior and motions to their behavior in real environments. By controlling the experimental factors, detailed comparisons of the effect of these environmental parameters and the differences between natural groups of pedestrians can be computed. There is evidence that this assumption is valid. Oxley et al. [Oxl+97] analyzed different response modalities to indicate a crossing with increasing complexity, from pressing a button over verbal indication to taking a single step forward. Participants accepted shorter gaps in the experiments compared to real environments. However, this difference was reduced with increasing complexity in the answer modality. While other studies utilized this knowledge to show a virtual character continuing the crossing with a constant velocity after two steps [SGS08], our virtual reality studies allow pedestrians to cross the road physically, adjusting their velocity and trajectory depending on the environment. Still, other factors like scene complexity [Oxl+97] or the underestimation of distances in VR [Plu+05; EM19] can impact the absolute difference to real behavior.

The virtual reality headsets used in this study have a visual field of $98°$ (Vive Pro Eye) and $110°$ (Vive XR Elite), while the natural horizontal field of view is $180°$ with a binocular overlap of $120°$. A loss of the central visual field (e.g., through scotoma, age-related macular degeneration, or simulations) can be compensated [AH20; HS12] but affects the head movements before crossing [Ger+06]. While the substantial loss of the peripheral field of view (visual field $< 20°$) has significant effects on crossing behavior [CGC08], the effects of a mild loss of peripheral vision are not sufficiently investigated. Future work will need to identify whether the inhibition of the field of view of virtual reality headsets influences crossing behavior or head movement.

However, a general experiment bias (e.g., more rule-following due to being observed) is most likely reduced due to the immersive headset. Participants neither see the lab nor the experimenters. Some were unaware of their location within the physical lab at the end of the experiment. All of these are, however, still assumptions. Future work is required to analyze the differences to real behavior and the overall validity of VR-based pedestrian behavior studies in large-scale environments. The best comparability would be achieved if virtual reality closely mimics the real environment in terms of its appearance and traffic dynamics. Using a physical driving practice area and instructed drivers would enable such high comparability. Conducting a study with the same participants taking part in the virtual and real trials and balancing the order of the trial groups would be an excellent basis for comparing the validity of behavior in virtual street environments.

Our feasibility study has shown that virtual reality experiments are feasible with school children. While there is a general question of whether preschool children accept virtual reality or the study environments at all, more playful scenarios could further reduce the minimal age. Utilizing virtual reality environments with child pedestrians opens up a range of topics future studies could investigate. The impact of body height, occlusions, distractors, and urgency could be safely investigated without harming a child. In addition, agents configured and trained to imitate this behavior would be valuable for evaluating and training autonomous vehicles in a digital reality. While a ball rolling on the street as an indication of a high likelihood of a child trying to catch the ball is a common challenge outlined in the press [Shw21], we are now able to investigate, simulate, and test this scenario and many other potentially dangerous situations (e.g., kids stepping out from school busses, illegal crossings, etc.).

Similary, the criticality of interaction can be increased in virtual experiments. While modulating classical factors like urgency is feasible [CFG12], the virtual environment offers more opportunities. For example, it is possible to generate new vehicles outside of the view frustum to model perceptual errors of pedestrians and trigger evasive reactions in near-crash situations. In addition to realistic near-crash motion, it could lead to the prediction of more realistic poses for in-crash simulations and passive safety analyses.

Besides simulation for the training of autonomous vehicles, the communication of autonomous vehicles with pedestrians can be investigated in virtual reality. Unlike hardware implementations, it is very affordable to develop and test virtual devices to communicate the behavior of the vehicle to the pedestrian, for example, light indicators, displays, or projective devices (see [RA19] for a review). While the visual quality and fidelity might not be strictly comparable to a real prototype because of the approximated rendering and missing global illumination used in VR simulations, it could be evaluated how pedestrians understand the semantics of the different communication devices in a highly controlled environment.

Chapter 4
Simulating Pedestrian Movements

The simulation of pedestrians in pre-crash situations requires the correct simulation of the crossing decision and the matching visualization through the body movements and the character animation. Thus, a multi-level multi-agent approach is beneficial, as the behavior simulation and the animation can be decoupled. In addition, each submodel of the pedestrian can be configured and trained based on the data observed in actual pedestrians, like in the experimental studies presented in Chapter 3. This chapter focuses on the simulation of the agent's body movements based on motion capture data.

Recent data-driven approaches using neural networks (e.g., [HKS17; Sta+21; Lin+20]) have shown impressive performance when generating locomotion movements, which are of primary interest for pedestrian simulation. However, for pedestrian simulation, it is not only necessary to train a single actor and produce walking animations. Natural differences in groups must be adequately represented to ensure fairness (e.g., cultural fairness, gender fairness, etc.). Models should not deterministically create animations but represent the natural variation of real people. On the more technical side, an agent model is supposed to control the animation, not a human player. Hence, the approach must be controllable and offer sufficient control vectors to be incorporated into different agent-simulation or crowd-simulation frameworks. In this chapter the contributions to generating pedestrian motion models are presented.

In Section 4.2, an approach towards simulating the natural inter-individual differences between female and male pedestrians is introduced. A linear model is trained to configure a phase-functioned neural network (PFNN) [HKS17] for a specific gender. The model is trained on motion capture data of 24 pedestrians (12 female) and the linear nature of the expressiveness of style was validated in a user study.

In Section 4.3, work towards generating intra-individual variation in locomotion is presented. The structure of a PFNN [HKS17] was extended with concepts of variational autoencoders to learn the variation in latent space. The resulting model can generate animations with an increased intra-individual variation without an increase in motion artifacts and errors.

In Section 4.4, efforts to excerpt more control over the motion based on mixture-of-expert models with explicit controllers are investigated. While the control of the gaze direction works sufficiently well, the direct control of other limbs (arms, legs) remains challenging.

In Section 4.5, a novel approach for animating avoidance movements of pedestrian agents inside of crowds is presented. Using a conditional variational autoencoder (cVAE) for animation and reinforcement learning (RL) for the controller, the pedestrian does not only avoid others but shows appropriate body movements while avoiding them (e.g., upper body rotations).

However, before going into the technical details, the existing work on motion synthesis is outlined in Section 4.1.

4.1 Related Work on Motion Synthesis

There are two different directions towards simulating the motions of a virtual pedestrian. On the one hand, **simulation-based methods** generate motion based on an internal body representation. The motion can be described by mathematical functions (e.g., [Kim14]) but more commonly relies on the actuation of a dynamic body model [Häm+14]. **Data-driven methods**, on the other hand, train a latent model on motion capture data and generate a simulation directly in the data space. As this dissertation focuses on data-driven methods, only those will be investigated in this section. We refer to the respective review papers for motion synthesis in general [WCW14] and for simulation-based methods in particular [GP12; Guo+15; Xia+17] for further information on the other methods.

Data-driven motion synthesis creates animation models based on motion capture data using kinematic animation data. While modern capturing equipment like IMU-based motion capture suits can create animation data of sufficient quality, capturing the motions itself is not the most time-consuming aspect. The captured motions need to be post-processed, cleaned, annotated, and cut to be utilized by the motion model. Games usually utilize blend trees, which require intensive effort, especially when complex animations and transitions between actions are required [Hol+20]. The ideal motion model, on the other hand, would be trained on unstructured, unlabeled motion capture data, would be able to generate a wide variety of controllable actions and natural transitions between actions with minimal adjustment effort, would be responsive to control and environment changes in a real-time fashion, and would have a fast computation time with a small memory footprint.

There are different topics relevant to data-driven motion synthesis and this dissertation: Sequence models generate full animation sequences, real-time motion models focus on responsive animations, statistical models try to replicate the natural distribution found in the data, style transfer is applied to change the distribution of a model to represent a certain style, and style learning tries to infer the style of natural groups (e.g., male and female pedestrians, Japanese and German pedestrians, children and adults, etc.) directly.

4.1.1 Sequence Models for Motion Synthesis

Sequence models can be defined as models that generate a whole animation sequence for a given control input (e.g., trajectory) or continue an existing motion sequence without external control (pose forecasting). Although lacking the responsiveness for real-time applications, they can achieve optimal solutions for a given constraint set.

Kovar et al. [KGP02] use motion clips from a dataset and build a *motion graph* encoding the different possibilities to reassemble the original clips. In this approach, a constrained graph walk generates target-specific animations based on the control input. Although the individual clips utilized as a basis for the graph are natural by design, the continuous animation can be distorted

by discretization, fast responses to a changing environment are impossible, and the approach does not scale to larger databases. There are multiple extensions for motion graphs. Lee et al. [Lee+02] apply unsupervised clustering of Gaussian mixture models (GMM) in addition to a Markov decision process modeling the transitions in a structure, very similar to a motion graph. Arikan et al. [AF02] create a *motion graph* on a complete motion sequence of consecutive frames to optimize the dynamics of a motion. The transitions in the graph are defined by annotated points in the time series. During inference, the graph walk generates a set of motion clips that must be blended to create the animation. Safanova et al. [SH07] extend the approach with an optimized searching function. By including joint torques in the objective function of the graph traversal planner, they increase the naturalness of generated transitions. Min et al. [MC12] provide a further extension to *motion graphs* by separating action and action variations. Walking, for example, can be considered as multiple transitions between left and right swing steps. The structurally similar motion clips are grouped into different motion primitives (left/right step), and a Gaussian mixture model is applied to learn the distribution for each primitive. A motion graph is built on top of the motion primitives to enable transitions between different steps. During inference, the authors employ a hybrid planning algorithm that firstly performs the graph walk operation similar to Kovar et al. [KGP02] and secondly uses probabilistic sampling with gradient-based optimization to find the closest motion clip for the next motion primitive based on the last motion clip. Du et al. [Du+16] optimize this approach by introducing scaled functional principal component analysis as a more efficient motion representation than consecutive Euclidean joint rotations.

There is a long history of neural networks being applied for motion modeling. Taylor et al. [THR11] extend a restricted Boltzmann machine (RBM) with autoregressive connections from past visible units (joint configurations) to the currently visible units as well as connections to the hidden units of the RBM. Further, they expand the model to multiple hidden layers. When trained on real motion capture data, the resulting model efficiently generates walking and running animations. Fragkiadaki et al. [Fra+15] use encoder-recurrent-decoder (ERD) networks trained on motion capture data. The encoder and the decoder are multi-layered, fully connected neural networks. The input vector is the joint configuration as joint rotations at frame f, and the network is trained to predict the configuration at frame $f + 1$. The authors recurrently apply the model to the output to generate motion sequences. Their network can robustly predict motion sequences up to 560 ms for walking motion. Li et al. [Li+18] use an auto-conditioned LSTM (long short-term memory) model, which is trained by alternating auto-recurrent predictions and ground truth values for the input of the network to combat the accumulation of error. Further, they include root velocity and use root-local joint positions instead of rotations as a motion representation. Their model can generate up to 300 seconds of random walk motion before divergence. Holden et al. [HSK16] build a latent space of motion by applying a convolutional autoencoder on a sequence of (positional) joint configurations. On top of the latent space, they train a feedforward neural network with additional abstract parameters, such as the trajectory (3D) of the root or the end effectors. As multiple motions can be used to follow a root trajectory, they train an additional third feedforward neural network to compute the footsteps out of a trajectory constraint. By applying multiple

constraints (e.g., joint positions, bone lengths, trajectory) in latent space, their model can generate a large variety of cyclic (e.g., walking) and noncyclic (e.g., punching) motions for multiple avatars in parallel.

With recent advancements in transformer and large language models, new sequence-to-sequence models for animation generation have emerged. For example, Li et al. [Li+21] generate a dance motion sequence based on an input seed motion and a longer music sequence with a cross-modal transformer-based neural network in an auto-regressive fashion. Ahuja et al. [AM19] propose a novel neural architecture based on GRU (gated recurrent units) and LSTM cells to learn a joint embedding space of motion and language that can be decoded to an animation sequence. During runtime, the proposed model can encode short text sequences to the latent space to generate corresponding animation sequences. Ghosh et al. [Gho+23] present a new architecture based on a pre-trained CLIP model [Rad+21] (Contrastive Language–Image Pre-training), multi-head self-attention and a fully connected architecture to generate the animation and object interaction based on a textual input (e.g., "take-picture") and the object (camera). Qing et al. [Qin+23] present a novel architecture using pre-trained LLMs to retrieve animation sequences from a database and neural motion blending to combine these sequences to long-duration animations matching a natural text describing a story.

4.1.2 Real-Time Motion Synthesis

Unlike sequence models, real-time motion synthesis focuses on responsive animation and causal single-step time-series models. The animation model predicts the next frame based on the control input and past animation for each frame update of the game engine. Hence, the compute time and responsiveness to changes in the control input are more important than finding an optimal solution. While some sequence models (e.g., [KGP02; AF02; Lee+10; HSK16]) can be adjusted for real-time synthesis, their solution usually lacks in quality of transitions, responsiveness, and computation time compared to models specifically addressing the real-time application. Models for real-time synthesis predict the single next frame based on the current and past states in addition to the user input. Thus, they are often combined with a separate motion controller maintaining the character state, incorporating the predictions, and the user control input (e.g., [Cla16; LLL18; Hol+20]).

Lee et al. [Lee+10] encode individual frames as motion states and model all possible transitions using a motion field rather than a graph. A Markov decision process (MDP) is built to model the possible actions for each motion state, and the reaction to a control input is enforced using a neural network trained with reinforcement learning. However, due to the k-nearest neighbors searching approach, the model does not scale, and its high computational complexity prevented the adoption in actual applications. Clavet et al. [Cla16] propose a simplification using a brute force approach (*motion matching*) that searches at every frame for the optimal next frame inside the motion database. Although not utilizing any internal model like a neural network, the approach produces natural-looking animations. It is widely adopted due to its implementational simplicity and low computational cost. Its major drawbacks, being poor scalability and high memory cost,

are tackled by Holden et al. [Hol+20], who introduce three neural networks: one to compress the database in a reduced latent space, one to predict the latent representation based on a state vector replacing the database lookup, and one to predict the next frames latent space for a frame step. Although the computation time is higher, their proposed approach drastically reduces memory consumption and scales well with new motion data.

Recurrent neural networks (RNNs) are specifically designed to model time-series data, as they incorporate the history of the already generated frames in their latent space. Some approaches apply RNNs to real-time motion simulation [Fra+15; Zho+18; LLL18]. However, the previous work shows that training RNNs is very complex, time-consuming, and uncontrolled generation of motion using RNNs suffers from dying out of motion and an accumulation of error [Fra+15; HKS17; Zho+18].

Simple regression models usually suffer from pose averaging due to similar intermediate poses in different pose sequences. For example, during a walk cycle, both legs cross. Due to missing information from the recent past, the network cannot distinguish which foot moves forward and does not accurately continue the walk cycle. Holden et al. [HKS17] propose a novel network architecture that interpolates the network weights depending on the phase of the walk cycle in a mixture-of-experts approach irradicating the pose averaging for locomotion tasks. This methodology is further extended by Zhang et al. [Zha+18] as a mode-adaptive neural network (MANN) to generate complex quadruplet motions of dogs. A proof of concept of the same idea for reach motions was published by Gaisbauer et al. [Gai+18]. Starke et al. [Sta+19] extend the approach to create animations with object interactions (e.g., sitting down, lifting a box, opening a door) by including environment sensors and extensive data augmentation. Although the animations appear natural, the accuracy is insufficient for high-detailed object interaction (e.g., finger dexterity). In addition, the approach does not scale well for complex and asynchronous multi-limb actions. Starke et al. [Sta+20] target the shortcoming in the simulation of fast and multi-limb correlated actions like basketball playing by extracting bone-based motion phases from the data and using a controller network mapping high-level controls to the low-level motion network input. The approach is further improved by Starke et al. [Sta+21] to allow explicit control of actions for the different limbs in a martial arts application (e.g., boxing and kicking) with a more elaborate, limb-based action controller network. The approach can generate close replications of specialized moves (e.g., spinning kick) and novel combinations of different actions (e.g., kicking and punching simultaneously). These approaches share the common feature that the regression network weights are interpolated in the latent space and a similar network structure for motion synthesis is utilized. In addition, these approaches require a complex, manually implemented motion controller that is highly optimized for a single application like walking, playing basketball, or boxing.

4.1.3 Statistical Motion Modeling

Neural networks tend to learn the mean motion of a dataset, primarily due to their deterministic nature. Most approaches use a mean-error loss function (e.g., mean squared error, L2-norm).

However, generating motions reflecting the statistical distribution within the data is important, especially for pedestrian simulation. Many conventional models have been used to model the motion distribution, for instance, variants of hidden Markov models (HMMs) [BH00; Bow16; LGN14], Gaussian mixture models (GMMs) in low-dimensional space [Lee+02; MCC09; MC12; Du+16] and Gaussian processes (GP) for temporal variations [LM07; WFH08; Zho+14].

Min et al. [MC12] combine a motion graph [KGP02] with statistical motion modeling. The motion data is modeled as a directed graph, each node being a cluster of semantically and structurally similar motion clips. Each edge in the graph represents the possible transition between clips. Motion clips in each motion primitive are projected into low-dimensional space, and a GMM is used to learn the distribution. Gaussian processes model the transitions between motion primitives.

4.1.4 Generative Neural Networks

Like statistical motion modeling, generative neural networks generate motion sequences matching a corresponding input (e.g., text, trajectory, action label) to generate new data similar to the statistics of the training dataset. Three major classes of generative methods can be differentiated:

- **Generative adversarial networks (GAN)** [Goo+20], in which a generator network and a discriminator network are trained in turns and in which the discriminator tries to distinguish the generated from the real data
- **Variational autoencoders (VAE)** [KW22], in which the encoder maps data points to a distribution within the latent space and the decoder generates data points based on samples of the latent distribution, making it a learned statistical motion model.
- **Diffusion networks** [Soh+15], a class of methods that aims to denoise data and can generate data points and sequences from a noise prior and a condition input, again being a stochastic model

Ahn et al. [Ahn+18] apply generative adversarial training to an RNN-based sequence-to-sequence model to train motion generation based on text input. While they can even apply the generative model to a Baxter robot, the authors only consider stationary actions and upper body movement. Wang et al. [WCX21] utilize an RNN to predict the probabilistic distribution of the next frame in a Gaussian mixture model. Adding a second RNN as a refiner model and applying generative adversarial training improves the realism measured in a user study and the quality and long-term stability of the generated motion. Degardin et al. [Deg+22] propose a spatio-temporal graph neural network trained in an adversarial setting. While the authors show a numerical improvement over past approaches for the application field and can generate 120 distinct actions, the motion quality does not match other state-of-the-art approaches.

Variational autoencoders (VAE) have recently been proposed as non-parametric, unsupervised approaches for manifold learning [KW22]. Since these models are non-parametric, no prior knowledge about data distribution is assumed. During training, the computational power of

neural networks is used to learn the transformation of an input from a normal distribution to any complex distribution. Holden et al. [Hol+15] apply convolutional autoencoders to learn a continuous manifold space for a large, heterogeneous motion database. Other solutions combine VAE with long short-term memory nodes (LSTM) [Hab+17] to model and predict motion sequences. However, these models cannot model a user's high-level control intuitively. Cai et al. [Cai+21] propose a new architecture to predict a masked or missing input of a motion sequence. They utilize a conditional variational autoencoder (cVAE) architecture, in which one encoder encodes the unmasked sequence while the other encodes the masked sequence. A combined decoder reconstructs the sequence, and a discriminator provides additional adversarial training. As convolutional layers are utilized, the whole sequence is generated in a single pass, and the model can be utilized for completion, interpolation, and future prediction. While the spatio-temporal reconstruction is especially compelling, the generative tasks result in jittery movements. Petrovich et al. [PBV21] utilize a transformer-based cVAE to predict complete motion sequences of varying lengths for a given action label (text). While their approach produces natural-looking animations of stationary actions, its output size is limited by the available hardware and can only generate non-interactive sequences. Zhang et al. [Zha+23a] utilize a vector quantized (VQ) variational autoencoder with temporal convolutional layers to encode motion sequences to a discrete latent space. A text-based transformer model is used to encode text into a latent sequence of these quantized tokens. The generative motion outperforms text-to-motion approaches but retains artifacts like jitter and foot-sliding. By utilizing a large language model (e.g., GPT-3 [Bro+20] or LLama [Tou+23]) as a text feature extractor, Zhang et al. [Zha+23a] can improve the quantitative measures further.

Diffusion models are a third class of generative approaches. Diffusion models are trained to solve the inverse diffusion process to denoise data from a random prior. In motion simulation, these models are primarily utilized for text-to-motion applications [Tev+22; Zha+24c; Zha+23b] and were extended to physical motion simulation [Yua+23]. In addition, diffusion models have been applied to other tasks, like audio-driven motion simulation [Ale+23]. Rempe et al. [Rem+23] utilize a diffusion model to generate trajectories of pedestrians. In contrast, the motion is generated by a physics-simulated character animated with a reinforcement learning controller to follow the trajectory. The diffusion model is trained on real trajectory data and generates avoidance and group behaviors. Natural crossing behavior could not be achieved, potentially due to missing data of real crossing scenarios.

4.1.5 Control Policies using Generative Networks

While there are powerful network architectures that generate natural-looking motion, the control of motion is equally important. Randomly generated pedestrians in an urban environment might lead to the most natural animations, but they are not necessarily useful for simulations. In most scenarios, pedestrians should stop at specific locations or look at specific directions at a well-defined point in time. The motion for these simulations must be controllable by an agent system and match

the observations of real pedestrians (e.g., gap acceptance, uncertainty behavior, waiting times, step-backs, etc.). Many neural-network-based animation models utilize hard-coded animation controllers to enable the walking direction or action (e.g., [HKS17; Sta+19; Sta+21]). However, a human player is assumed to provide the control input in these applications, not a virtual agent model. While the human player automatically corrects any bias or error of the animation controller and often does not care about reaching a specific target with high precision, more accurate control is required for pedestrian agents to prevent unwanted behavior – like stopping too late and stepping on the street in a scenario where the pedestrian should remain on the sidewalk. In addition, the controllers are implemented for single tasks (e.g., following an input direction, punching an opponent with the right hand, etc.), do not generalize well, and require extensive programming (e.g., over 10,000 lines of C# code for the animation controller in [Sta+20]).

A different way to approach the animation controller is with reinforcement learning, training a neural network to provide the control input for a different neural network generating the motion data. Ling et al. [Lin+20] utilize this concept by training a cVAE based on a recurring mixture of expert architecture (motion VAE). As an input for the decoder, the past frame and the latent control sample are passed through a MANN-style network [Zha+18] to generate the next frame. A reinforcement learning controller is trained to generate the control input for different locomotion tasks (target, path following, joystick control, maze running). While the results are very promising, we found that the input data distribution, the hyperparameters of the cVAE (e.g., KL-loss penalization), and the reward function must be carefully selected to generate similar results. Shi et al. [Shi+24] apply reinforcement learning to similar tasks based on an auto-regressive motion diffusion model, achieving higher diversity and faster computation time than motion VAEs [Lin+20]. However, the generated motion contains severe artifacts (e.g., foot sliding) for some of the tasks. Starke et al. [Sta+23] develop a novel periodic autoencoder, which compresses motion to a phase manifold, a set of multiple two-dimensional vectors that retain the spatio-temporal relationship of the motion inside the latent space. Due to the shape of this latent space, it is a good candidate for a different latent representation to be controlled by reinforcement learning.

4.1.6 Style Transfer of Artificial Style Examples

There are different approaches to generating motion in a specific style. One of the most common approaches is style transfer or style translation [HPP05]. A "neutral" content motion prescribes an action (e.g., run), which the avatar should perform, while the style example defines the (artificial) style in which this action should be performed (e.g., sexy walking). Hsu et al. [HPP05] use a linear time-invariant system to translate style within the same action class. Styles and actions, however, are not always decoupled. Ma et al. [Xia10] consider style in locomotion as the stride length and velocity, whereas variation denotes dynamic differences between actors. Brand et al. [BH00] use the term "style" to describe different gait types (walking, running, etc.) and the term "expert level" to differentiate dance motions. It is important, however, to decouple style and action. In a controlled motion synthesis, the user wants to define the action type (e.g., walking to running). In

contrast, the style is defined by the character (e.g., female or male character) and its mental state (happy, sad, etc.). Min et al. [MLC10] specifically address this problem and model different style and identity variations for multiple actions using motion capture data containing artificial styles (angry, tired, etc.) and a multilinear motion model.

Xia et al. [Xia+15] published a large dataset containing multiple actions (walking, running, punching, kicking, etc.) and a methodology that can transfer style over homogeneous and heterogeneous motions in real-time. The authors use local mixtures of autoregressive models and a nearest neighbor search to find mixtures from the source database to transform a single frame. Yumer et al. [YM16] performed a user study using Xia's dataset and report confusion of "sexy" with "proud" and "depressed" with "old". Thus, although expressive, these artificial styles are just a single example and cannot automatically be considered as a natural archetype.

Holden et al. [HSK16; Hol+17] utilize techniques used for style transfer in an image domain by first, optimizing the hidden units of an autoencoder to match the Gram matrix of a stylized motion example [HSK16] and second, by using an additional loss network to speed up the process [Hol+17]. Mason et al. [Mas+18] apply transfer learning to fine-tune the network weights of a phase-functioned neural network [HKS17] to a specific input style and thus create style-specific network representations.

The approaches discussed so far are transferring style between motions with increasing performance. They do not learn the style itself but require exemplary clips for a specific style and try to translate or replicate this specific style. In many cases, they are optimized for the stylization of animation as required by gaming applications (e.g., walk like an orc) and are only partially applicable to pedestrian simulation. While orcs are rarely seen in urban environments, other very distinct walking styles for which data capturing is inherently difficult are clinically relevant gaits (e.g., Parkinson's disease).

4.1.7 Natural Style Learning

Given a larger dataset containing different subjects of different natural groups (e.g., gender, age, weight, cultural background), it is beneficial to learn the distinct style of each natural group directly. McDonnell et al. [McD+09] analyzed the influence of body shape and dynamic motion on the perception of the character's gender. In the case of neutral walking motions, the perceived sex can be attributed to the body shape. However, when the motion of male or female characters is mapped to either androgynous or realistic body shapes, the response is affected considerably. Hence, the generated dynamic motion should match the avatar's body shape to create a coherent and plausible virtual character.

In cognitive psychology, humans' perception of biological motion [Tro13] was analyzed excessively in the past decades. Johansson et al. [Joh73] display motion as point-light walkers, which can be accurately perceived as human motions (see Figure 4.1). Kozlowski et al. [KC77] show that human observers can recognize the gender of a walker displayed in a side-view with an accuracy of 63%.

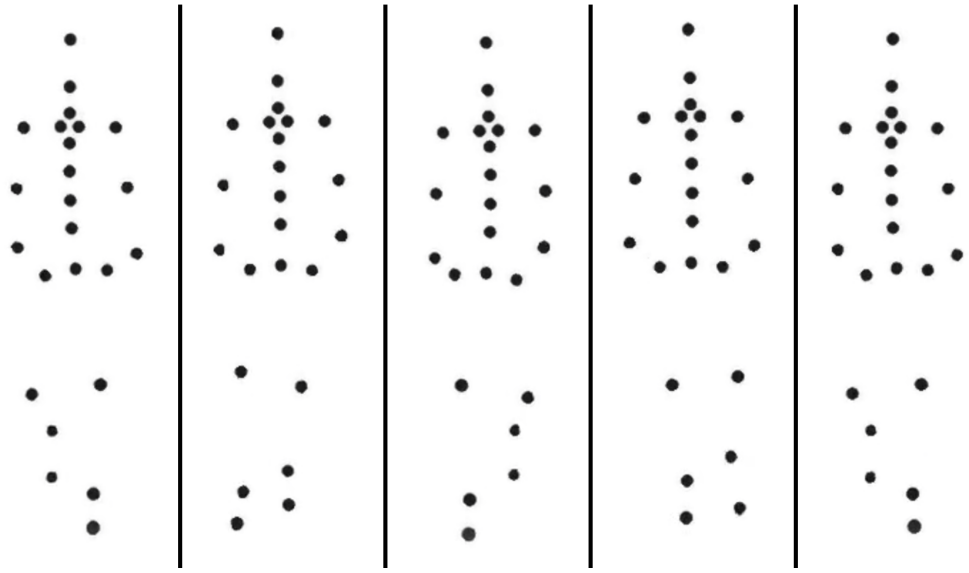


Figure 4.1: Point-light-walker sequence for female walking using a linear style model. Images are taken from a video sequence in the order of left to right. The point-light walkers are displayed with inverted colors. Reprinted with permission from [Spr+19b].

Troje et al. [Tro02] decompose motion into static (body shape) and dynamic (motion) information. The cyclic walking motions are decomposed with a PCA, and the difference in the walker-gender is modeled with a linear regression. The resulting interpolated motion was accurately recognized in a user study. Dynamic motion mapped to the average point-light-walker was recognized more accurately as the dynamic motion mapped to the respective stylized point-light-walker. Recognition rates were better in the full frontal than the side view. The gender of examples using dynamic-only information was recognized with 75% accuracy.

Besides directly learning the style from data in a bottom-up process, there is some effort in defining a style from a top-down perspective. For this purpose, the Laban movement analysis (LMA) can be utilized. LMA is a movement annotation system that classifies qualitative and quantitative human movement characteristics. It can be used by human observers to identify the systematic differences between different motion styles. For example, Chi et al. [Chi+00] utilize LMA to synthesize gestures. Durupinar et al. [Dur+16] use LMA experts to specify several style parameters influencing a motion clip to generate motion corresponding to the personality traits defined in the Big Five personality model [Gol90]. Torresani et al. [THB07] assign LMA-Effort labels to motion clips of dance performances to transfer the individual style between dancers.

4.2 Learning and Simulating Gender-Specific Motions

To ensure that pedestrian agents are fair (e.g., culturally fair, gender fair, etc.), it is important to create agent models that adequately reflect the intrinsic parameters of these natural groups. This section focuses on the locomotion style, specifically gender differences. Previous work on stylistic motion synthesis either considers **artificial style** (e.g., sexy, old, tired, etc.) [MLC10; Xia+15; Cha+19] or **individual style** [THB07; McD+09].

An **artificial style** is played by a trained or untrained actor in a predefined manner, usually without further analysis of the human perception of the style. These styles are usually very exaggerated but not necessarily natural. For example, the "sexy walking" from the dataset published by Xia et al. [Xia+15] depicts a catwalk-like walking pattern that would be attributed to a female actor. It shows a large hip-sway and very characteristic arm movement and foot placement. However, this style is not the natural style of a female pedestrian. Yumer et al. [YM16] report a confusion of "sexy walking" with "proud walking" in a recognition task using this specific motion capture data, thus further undermining the validity of style labels.

Work on **individual style** considers each actor as an example of a higher-order group (e.g., male and female actors). The individual style is an example of this group and should be recreated or transferred. Approaches aim to utilize natural data as a basis, reducing the effects of recording, having uninformed participants, and utilizing habituation intervals [Tro02; THB07; McD+09]. Although the style is natural, (motion) actors usually create a more distinctive, exaggerated, and artificial style. McDonnell et al. [McD+09] report that the perceived intensity of a style can vary between actors when queried in perception experiments. For this dissertation, learning and generating these inter-individual style differences is of interest.

The animation style is very important in humans' perception of humanoid avatars (e.g., human drivers). McDonnell et al. [McD+09] could show that the perceived sex of virtual characters changes with the animation style. A model for stylistic motion synthesis should not only be able to differentiate between male and female locomotion but also to vary the intensity to steer the perception of a character. As Chan et al. [Cha+19] point out, style is usually labeled in a binary fashion rather than differentiating between the intensity of style. They propose to use an additional preprocessing step to rate this intensity. However, Troje [Tro02] analyzed the body movement of female and male participants. His approach can differentiate the motion and synthesize it with differently perceived intensity without additional labeling. His approach, however, only operates on very simple, periodic walk cycles on a straight line and thus is not flexible enough to satisfy more complex application fields. The aim of stylistic motion synthesis should be to synthesize binary style differences and to vary the expressiveness of a specific style. Style can be considered on a spectrum or a continuous space, and thus, a stylistic motion model should not only generate a single example from this spectrum.

We propose a data-driven approach to learning the style from multiple individuals performing natural walks rather than from an individual actor playing different intensities of the style. It

learns style differences in latent space and generates complete motion models rather than directly synthesizing them. Unlike models for artificial style, our approach does not utilize single examples played to appear in the respective style (male walking, female walking) but examples of the natural motion of individuals from the respective sex groups. Hence, by capturing their fundamental differences, our model can learn the stylistic differences between two natural groups based on examples only. Using state-of-the-art neural networks for real-time motion synthesis enables controllable animation such that complex locomotion tasks (walking, running, stopping, turning, etc.) can be accurately synthesized.

A total of 112 minutes of motion capture data for locomotion actions of 24 participants (12 female) was captured in a controlled experiment, presented in Section 4.2.2. The generated animation of our model can create styles with different intensities. The resulting motion was evaluated in a human perception experiment to support the validity of the generated style and highlight the shortcomings of naïve approaches. The details and results of the evaluation are presented in Section 4.2.3. The discussion of our results and the implications for future work is highlighted in Section 4.2.5.

4.2.1 Approach

During neural network training of the motion models, the optimization algorithms adapting the network weights are designed to abstract from individual differences between the samples and try to optimize the prediction performance of the network for all samples. Hence, when training a network on a balanced motion capture dataset of different individuals, the network will learn the general concept similar to all individuals rather than the individual characteristics by design. Training the network only on the motion capture data of one individual will replicate its characteristics. Considering the work of Troje [Tro02], it should be possible to discriminate the features for higher-level characteristics, for example, gender-specific differences, inside the latent level of the motion model. We first present the data model used to train the base motion model synthesizing the animations, which is presented afterwards. The linear style model, which generates gender-specific motion models, is presented next, followed by information on model training and different alternative models, which are compared.

Data Model. For the motion synthesis, the motion capture data is processed to extract different types of information for the animation pipeline, including the root trajectory and the gait information. For a single frame, the data is considered in the root local coordinate system, which has its origin at the projection of the hip onto a virtual ground plane oriented towards the forward-facing direction of the character. The root trajectory is sampled at six points in the future and six points in the past with a spacing of ten frames between every sampling point. The trajectory positions ($t^p \in \mathbb{R}^{24}$) and the trajectory directions ($t^d \in \mathbb{R}^{24}$) local to the root position and direction are incorporated, as well as the gait labels at each trajectory point ($t^g \in \mathbb{R}^{36}$). The network output contains only the future trajectory points for the position $t^p_{i+1} \in \mathbb{R}^{12}$ and directions $t^d_{i+1} \in \mathbb{R}^{12}$.

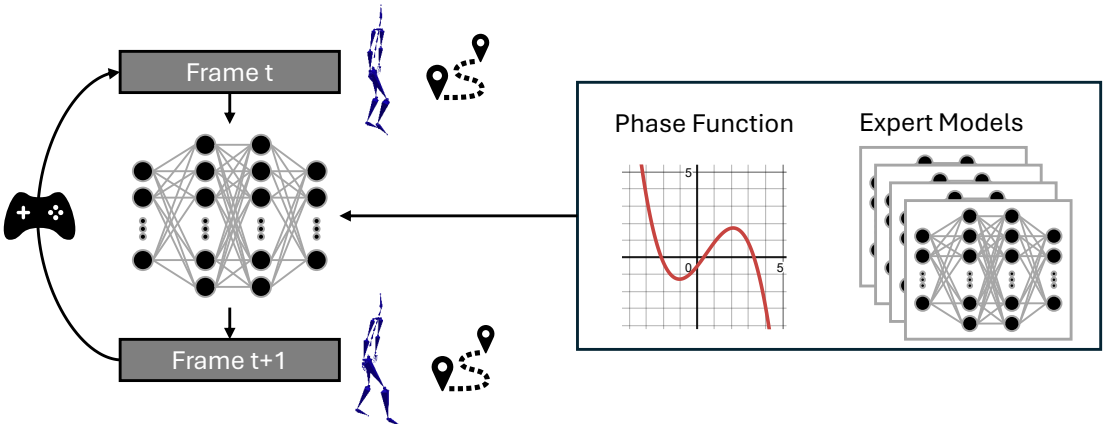


Figure 4.2: Concept visualization of a phase-functioned neural network. The phase function is a cubic Catmull-Rom spline, which uses the motion phase as an input to combine the weights of four expert models to the motion model in every frame. An animation controller maintains the character's state, incorporates user input, and uses the network prediction to create the animation.

The pose is encoded as a set of joint positions $j^p \in \mathbb{R}^{78}$ and joint velocities $j^v \in \mathbb{R}^{78}$ with both including 21 joints and five end nodes (e.g., top of the head). The bone rotation $j^r \in \mathbb{R}^{21}$ is reduced to the twist of the bone along its forward axis. For this purpose, the bone rotation is separated into the swing component (rotation towards child joint/end node) and twist component (rotational component around the bone axis) using swing-twist decomposition [Dob15]. Hence, end nodes are added to the joint positions and velocity values as virtual joints. During inference, the joints are first rotated such that the forward axis points toward the child's joint, and afterward, the twist is applied. For the output of the prediction, additional root velocities projected on a virtual ground plane $(r_i^x, r_i^z) \in \mathbb{R}^2$, the angular velocity of the character root around the up axis $r_i^a \in \mathbb{R}$ as well as the change in phase $p_i^\Delta \in \mathbb{R}$ are incorporated. The full input vector x_i and output vector y_i used for a single regression step are thus:

$$\begin{aligned} x_i &= \left\{ t_i^p \; t_i^d \; t_i^g \; j_{i-1}^p \; j_{i-1}^v \; j_{i-1}^r \right\} \in \mathbb{R}^{261} \\ y_i &= \left\{ r_i^x \; r_i^z \; r_i^a \; p_i^\Delta \; t_{i+1}^p \; t_{i+1}^d \; j_i^p j_i^v \; j_i^r \right\} \in \mathbb{R}^{205} \end{aligned} \tag{4.1}$$

Base Motion Model (PFNN). In this work, phase-functioned neural networks (PFNN) [HKS17] are used as a motion model, which a specialized form of mixture-of-expert models. Hence, the information of a single model is encoded in the parameters of the (hidden) layers of the network. Using a cubic Catmull-Rom spline (see Appendix A.1.3), the parameters of four expert models are combined into the effective motion model for a single inference step. The phase during a locomotion cycle (from one left stance to the next left stance) is used as an interpolation weight. The output vector is passed to a motion controller, which maintains the current state of the character and incorporates user input (movement and heading direction). The motion model, as well as the individual expert models, is a fully connected three-layer neural network using the exponential

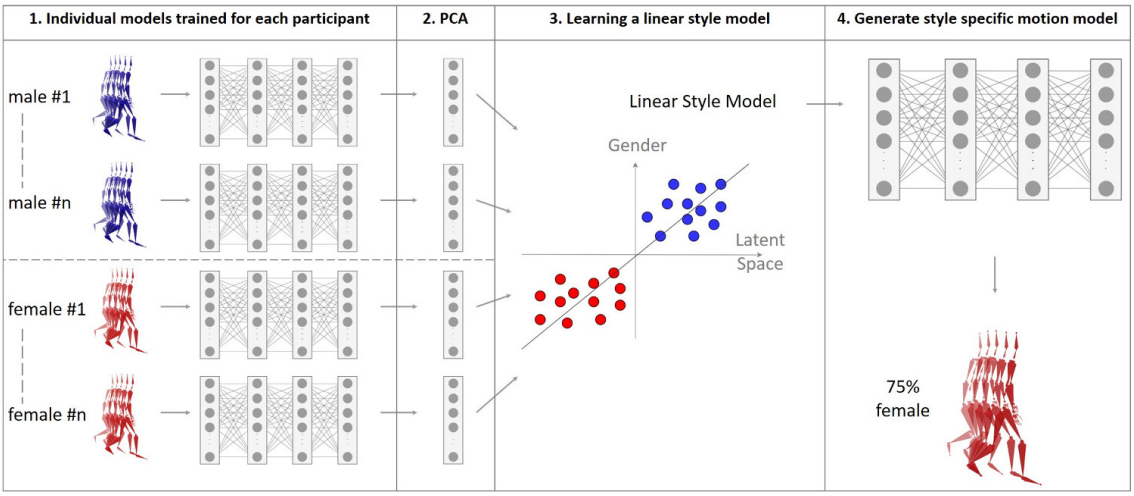


Figure 4.3: Visual representation of the linear style model (LSM) using PCA decomposition. Individual network models are trained for each participant, and the dimension of each layer is reduced with a principal component analysis. A linear regression is fitted to the reduced space and can synthesize networks generating motion with a specific expressiveness of style. Reprinted with permission from [Spr+19b].

linear units (ELU) (see Appendix A.1.1) as an activation function. Figure 4.2 shows the conceptual function of the motion model. The dimensionality of the hidden layers is 512. Thus, the total full network has $4 \times 545,015$ parameters. Using these parameters (weights W and biases b) of the network, it can be described as:

$$\Theta(X, W, b) = W_2 \times ELU(W_1 \times ELU(W_0 \times X + b_0) + b_1) + b_2 \tag{4.2}$$

Linear Style Model (LSFM). Similar to Troje et al. [Tro02], we compress the individual parameters of the expert models of the PFNN using PCA and utilize a linear regression model to differentiate gender. A visual representation of this approach can be found in Figure 4.3. Our **Linear Style Model (LSM)** is generated with the following steps:

1. Training of N networks for N individual persons inside the motion capture dataset and computation of the residual walker models.
2. Dimension reduction using PCA is applied on each network layer separately.
3. A linear regression is fitted for each network layer separately.
4. Online generation of networks using the linear regression as a generator and transformation to the high dimensional latent space.

There must be a correspondence between weights to interpolate the models [BV99]. The networks of two individuals displaying a very similar walking pattern should be very close, and all dissimilarities inside the network weights should only arise from the differences in walking style. One straightforward approach to generating this correspondence is transfer learning, commonly used for neural networks to reduce training time and boost performance. In this case, the whole motion

capture dataset can be used as an input to train a general knowledge model (GKM) for walking. This model is used as a basis to fine-tune N models for each individual participant (walker models). With the parameters for the training set sufficiently well, there is enough correspondence between networks to apply a discriminator in the hidden space. The parameters of all N individual walker models are averaged (average walker), and the residual weights of the individual walkers are computed via subtraction of the average walker (residual walkers). PCA is applied to the parameters of the individual layers of the residual models to reduce the dimensionality to the four most relevant components (20% - 80% variance retained). A linear regression model is fitted to this reduced space to differentiate male from female residual models on a per-layer basis using the scikit-learn library [Ped+11]. The linear model can be used to interpolate between two styles but also to extrapolate (absolute interpolation factor > 1). Hence, the term "style configuration" is used in the following.

The linear regression model generates style-dependent motion models in the compressed parameters of a residual walker model. After uncompressing, the average walker model is added, and the full parameters of a PFNN are restored. This gender-specific PFNN is utilized afterwards to generate the pose. For interactive motion synthesis, a similar controller as proposed by Holden et al. [HKS17] is implemented in Python. The generated motions are streamed over a network connection and visualized in Unreal Engine 4.

Model Training. All models presented in this work are developed using the TensorFlow [Aba+15] framework and are trained using the stochastic gradient descent algorithm Adam implemented in TensorFlow. Dropout was applied with a retention probability of 0.7. All models are trained in mini-batches of size 32. The general knowledge model (GKM) was trained for 60 epochs with a learning rate of $\beta = 10^{-4}$ on the whole training data of all participants. Based on the general knowledge model, individual walker models, as well as the gender-specific archetypal models, have been trained for an additional 20 epochs with a reduced learning rate of $\beta = 10^{-5}$ in order not to deviate too far from the general knowledge model.

Model Comparison. Besides the approach outlined above, there are multiple ways to apply the linear model. All variables can be combined in a single vector, and the regression can be computed using this vector. Pretests showed that these global models could not generate visual or numeric differences. However, there was a numerical and visual difference when performing the linear regression on each layer (network weights and biases) separately. By applying principal component analysis to reduce the dimensionality of each layer first before applying the style regression, we could increase the stability of the generated motion.

The presented approach of **LSM** is compared against three baseline methodologies. A vanilla phase-functioned neural network using participants' gender as a separate network input was trained and evaluated as a baseline model. This model is subsequently called **VANILLA**. In traditional motion capture datasets, there are usually performances of single, trained actors displaying an artificial style. To mimic this setting, two archetypal style models (ASM) were trained for all male and all female participants. These two networks were combined using the same approach as the LSM and

Figure 4.4: A photo of the experimental space. The capture area was encircled with a barrier tape to prevent participants from stepping out of the reserved space.

are subsequently called **ASM-PCA**. In addition, we applied direct linear interpolation between both networks on the latent level without training a classifier and without dimension reduction with PCA. Subsequently, this model is called **ASM-LIN**. Initial tests showed, however, that this approach was not stable under extrapolation and created a high amount of visible artifacts.

The body parameterization (e.g., height) was adjusted to each participant individually, resulting in structural differences in the scale of data captured with different participants. During model training and motion synthesis, these body configurations can be considered using proper motion parameterization. Hence, a second level of complexity is analyzed by comparing models trained on motion capture data aligned to the average hip height (**aligned** data) with models trained on motion capture data retargeted to the average skeleton (**normalized** data). For each configuration, two models were trained using the same random split between training and testing data: one using normalized input data and one using aligned input data. Thus, each pose in the normalized set had the equivalent aligned pose in the aligned dataset.

4.2.2 Data Capturing

Capturing Procedure. In a controlled experiment, the locomotion data of 24 healthy participants was gathered using the IMU-based motion capture suit Perception Neuron. The data was captured on a 4.5 m x 12 m flat surface in front of the research facility. A roof over the area and a building to the west provided some shadow during midday and the afternoon. Figure 4.4 shows a photo of the capturing area. Capturing took place in daylight between 8 a.m. and 6 p.m. All participants were recruited from the university's student body. The participation was voluntary and not monetarily reimbursed. No participant has performed with a motion-capturing device before. The study was approved by the Ethical Review Board of the Department of Computer Science (No. 19-1-4).

During the experiment, participants moved around the area at a preferred speed and direction in

Table 4.1: Descriptive statistics for all participants containing the mean values for each gender group. The asterisk * denotes a significant difference ($p < 0.01$). Reprinted with permission from [Spr+19b].

Group	N	Age	Height	Weight
Male	12	26.67 yrs	179.58 cm *	76.25 kg *
Female	12	24.67 yrs	166.17 cm *	60.58 kg *

six trials of about one minute. Each trial consisted of 13 blocks lasting for about five seconds each. At the beginning of each block, a target gait was announced verbally, and the participant had to change their gait in a controlled manner. The target gaits were standing, walking, running, and backwards walking. Initial tests revealed a preliminary or abrupt change of gait if the commands were provided in a regular beat, hence the exact timing of the command onset was randomly varied. The first training trial was not recorded and was used to accustom the participants to the motion suit and the experimental situation. By design, the first three blocks of each trial were excluded from the evaluation, as unnatural motions in these blocks were expected due to discomfort and an unnatural voluntary focus on walking by the participants.

A major drawback of IMU-based motion capture solutions is the lack of global translation. The software provided with the motion capture solution can generate relative motions but only partially reconstruct the global trajectory and translation. Hence, a multi-step approach was developed to find footstep locations robustly using advanced machine learning approaches [Nos+18]. The footstep locations can be fixed to the ground by inversely applying the foot-sliding to the global root translation. After post-processing, all motions were mirrored by switching the rotations of left and right limbs and flipping all rotations, thus effectively mirroring the whole motion around the global up-axis. The aim of this step was not only data duplication but also the attempt to balance the dataset concerning the walking direction. If there is a bias inside the dataset of turning more to the left than the right, mirroring the data balances this directional bias.

Dataset Analysis. We statistically analyzed the participants' characteristics and the captured dataset. All statistics were computed using the software package IBM SPSS, Version 25 [IBM17]. All variables were tested for normal distribution using the Shapiro-Wilk test [SW65] for pairwise comparisons. When the assumption of normal distribution did not hold, non-parametric tests were used. The same holds for the assumption of the equality of variances, which was tested and corrected using Levene's test [Lev60] when required. In this work, effect sizes are not reported, as they can be computed trivially [Ras+14, p. 129].

The descriptive statistics are reported in Table 4.1. An independent samples t-test confirms a difference in body height, $t(22) = 4.659$, $p < 0.001$, but not in age, $t(22) = 1.88$, $p = 0.073$. A Mann-Whitney test confirms a significant difference in body weight, $U = 18$, $p < 0.01$.

On average, 35,091 frames per participant were captured, and the whole dataset comprises 112 minutes of motion-capturing data. The exact frame numbers for each participant are within three standard deviations around the mean.

Table 4.2: Average velocities of groups for each gait. The velocity significantly differs between gaits but not between gender groups. Reprinted with permission from [Spr+19b].

Group	Backwards	Walking	Running
Male	1.01 m/s	1.21 m/s	2.34 m/s
Female	0.94 m/s	1.23 m/s	2.24 m/s

The difference in locomotion velocity between gaits and gender is analyzed using a univariate analysis of variance (ANOVA). A `Gender` (2) x `Gait` (3) repeated measures ANOVA is applied. Table 4.2 shows the average velocities for each gait. The results show a significant main effect of `Gait`, $F(2, 44) = 1120.7906, p < 0.001$, but do not confirm a significant interaction effect. Estimated marginal means and pairwise comparison show a significant difference in velocity between all three gaits for both genders (all $p < 0.001$). Hence, locomotion velocity significantly differs between gaits (e.g., between walking and running) but not between male and female participants.

4.2.3 Human Evaluation

A human perception study was conducted to evaluate the naturalness of the generated motion. In this study, the participants observed short clips (seven seconds) of motion in a point-light walker display showing walking and running motions from different approaches. They had to rate the gender of the displayed "actor". Example pictures of a motion sequence generated by an LSM model in this point-light walker can be seen in Figure 4.1. The experiment was designed similarly as reported by Troje [Tro02], where classification performance of 70% - 80% were reported. As it poses a binary decision problem, the chance level is at 50%. The main hypothesis of this evaluation study was that the human observers were able to estimate the visualized gender better for motions generated with LSM than with the other methods.

Method

Participants. A total of 21 participants (12 female) participated in the evaluation experiment. The descriptive statistics are in Table 4.3. All participants were recruited from the university's student body. The participation was voluntary and not monetarily reimbursed. No participant was experienced in motion synthesis or point-light displays. This study was evaluated and approved by the Ethical Review Board of the Department of Computer Science (No. 19-1-3).

Material. The motion was visualized using point-light displays [Joh73]. A single virtual ball of light was positioned at each joint and the end-effectors (hands, feet, head). A total of 24 virtual

Table 4.3: Descriptive statistics for the human evaluation experiment. Reprinted with permission from [Spr+19b].

Gender	Participants	Mean Age	Std. Dev
female	12	23.75	0.24
male	9	25.56	0.26

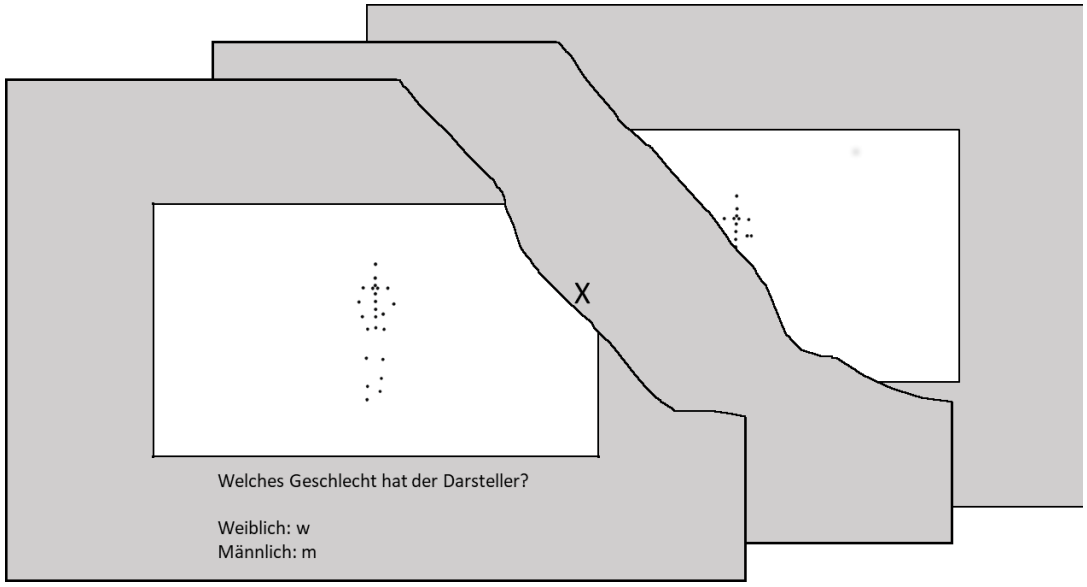


Figure 4.5: Procedure of the human evaluation. Video stimuli are presented in inverted colors in this document. Reprinted with permission from [Spr+19b].

light bulbs compose the displayed pose. Figure 4.5 shows an example of the configuration. Each clip shows a single gait for a single style configuration for seven seconds. Two clips were generated for each of the gaits, walking and running, for interpolations between -2 to +2 with increments of 0.5. The extrapolation was excluded for both archetypal networks using direct linear interpolation, yielding ten clips each. Hence, a total of 128 clips were presented to the participants. The experiment was implemented and executed using PsychoPy V1.90.3 [Pei09].

Models trained on aligned data can reproduce the structural difference between male and female participants. In contrast, models trained on normalized data cannot reproduce static differences by design. To accurately compare both methods, we implemented a dynamic-only display. The static structure of the point light walker corresponded to the average skeleton used for dataset normalization. All bone lengths (distance between joints) were restricted, including the hip and shoulder width. The motion generated by the different models was mapped to this skeleton in the simulator. Hence, the only difference between the two clips was the dynamic part of locomotion.

Procedure. The displays were positioned one meter from the participants, and the figure was presented on the display at a physical height of 12 cm. Participants could adjust their seats as required. Every 32 trials or 5-6 minutes, there was a short break to prevent fatigue. The trials were fully randomized across all blocks.

Each trial started with a fixation cross displayed for three seconds to enable the user to differentiate between trials. After the fixation cross, a video stimulus of a walking animation as a point light display was presented for seven seconds. The participants could provide an answer during and after the video presentation without any time restrictions. The video was not skipped after the answer was provided to prevent participants from reducing experiment time. Figure 4.5 provides an example of this procedure.

Before the actual experiment, four training trials were conducted to accustom the participants. Each training trial was repeated until the participant could rate the gender of the displayed walker correctly. The trials showed animations for each gender and gait. They were generated with the basic gender networks of the archetypal network trained on aligned data, with the hip- and shoulder constraints disabled. Hence, female clips displayed broader hips and narrower shoulders than male clips. This reduced the complexity of the task compared to the experimental trials.

Statistics. The mean accuracy of the responses of each participant was evaluated for each `method` (4), training dataset (aligned, normalized), and `gait` (walking, running). A repeated measures ANOVA was applied to analyze the difference in response accuracy between all variables. Post-hoc tests were used to analyze the difference further. For multiple comparisons, all p values are corrected for multiple comparisons using Bonferroni correction. All statistics were computed using the software package IBM SPSS, Version 25 [IBM17]. All variables were tested for normal distribution using the Shapiro-Wilk test [SW65] for pairwise comparisons. When the assumption of normal distribution did not hold, non-parametric tests were used. The same holds for the assumption of the equality of variances, which was tested and corrected for using Levene's test [Lev60] when required. In this work, effect sizes are not reported, as they can be computed trivially [Ras+14, p. 129].

4.2.4 Results

The main results of our user study are:

- Only the stylistic differences of LSM and LSM-LIN trained on normalized data are recognized above the chance level.
- Neither adding the style as a parameter to the network input (VANILLA) nor ASM-PCA models generate a motion with distinguishable style differences.
- Motion generated with LSM models is recognized significantly better than motion generated by any other method.
- Although there is no statistical difference between LSM models trained on aligned or normalized data, the style generated by the LSM aligned model is recognized best with an accuracy of **75.6%**.

The average rating for the different interpolation factors of motions generated by these models is displayed in Figure 4.6. The answers were encoded with -1 for male and +1 for female response. Inside the graph, the results of models displaying a style classified above the chance level are highlighted in solid colors. The average answer does not correspond to the style configuration for most of the evaluated methods. A linear trend can be observed only for linear style models (LSM) and the direct interpolation of archetypal networks (ASM-LIN). This supports the assumption that the LSM models encode the stylistic differences between female and male participants and can generate motion models displaying this style.

Table 4.4: Results of the human evaluation for the different evaluated models. Results are reported as mean classification accuracy over all gaits and interpolation factors. The chance level is 50%. Classification scores that are significantly different from chance level are marked with an asterisk (p < 0.001: **, p < 0.05: *). Reprinted with permission from [Spr+19b].

Method	Dataset	Accuracy
LSM	aligned	75.60% **
	normalized	73.81% **
ASM-PCA	aligned	50.89%
	normalized	52.38%
ASM-LIN	aligned	58.93%
	normalized	61.90% *
VANILLA	aligned	50.30%
	normalized	55.65%

Subsequently, the detailed results of our statistical analysis are reported. The `Method` (4) x `Dataset` (2) repeated measures ANOVA reveals a significant main effect of `Method`, F(3, 60) = 36.165, p < 0.001. Post-hoc analysis using a one-sample Wilcoxon signed rank test shows that only motions generated by three models were classified correctly above chance level (50%). LSM trained on aligned data ($Z = 3.974, p < 0.001$), LSM trained on normalized data ($Z = 4.029$, $p < 0.001$) and ASM-LIN trained on normalized data ($Z = 2.732$, $p = 0.048$). The synthetic motion generated by all other models was not classified correctly above chance. The nonparametric test is used because variables partially violate the normality assumption. However, an analysis using one-sample t-tests reveals the same results. The mean accuracy for all models and datasets used in this evaluation are displayed in Table 4.4.

Additional post-hoc tests using estimated marginal means show a significant difference between the classification of motion generated by the LSM models and the classification of motion generated by all other models for both `Dataset` conditions (each $p < 0.01$). Neither do the other methods differ from each other, nor do LSM (normalized) and LSM (aligned) differ significantly.

4.2.5 Discussion

Our user study shows that the motion generated by LSMs can be classified by human raters in a comparable performance as reported in previous publications [Tro02]. In addition, the configuration of the linear model directly affects participants' answer rates. For example, The more a network is configured to represent female walking, the more participants rated the generated motion to be from a female actor (see Figure 4.6). This effect is stable for extrapolation, showing the capability of the presented approach to intensify the style above the mean. Unlike previous work on style transfer [Xia+15; YM16], our method can learn the continuum of style expressiveness directly from unstructured motion capture data. Example visualizations of motion sequences generated for female and male walking with the LSM aligned model can be found in Figure 4.7.

If trained on a balanced dataset with equal samples for male and female participants, an LSM configured with a style configuration of zero should create a realistic neutral walking animation.

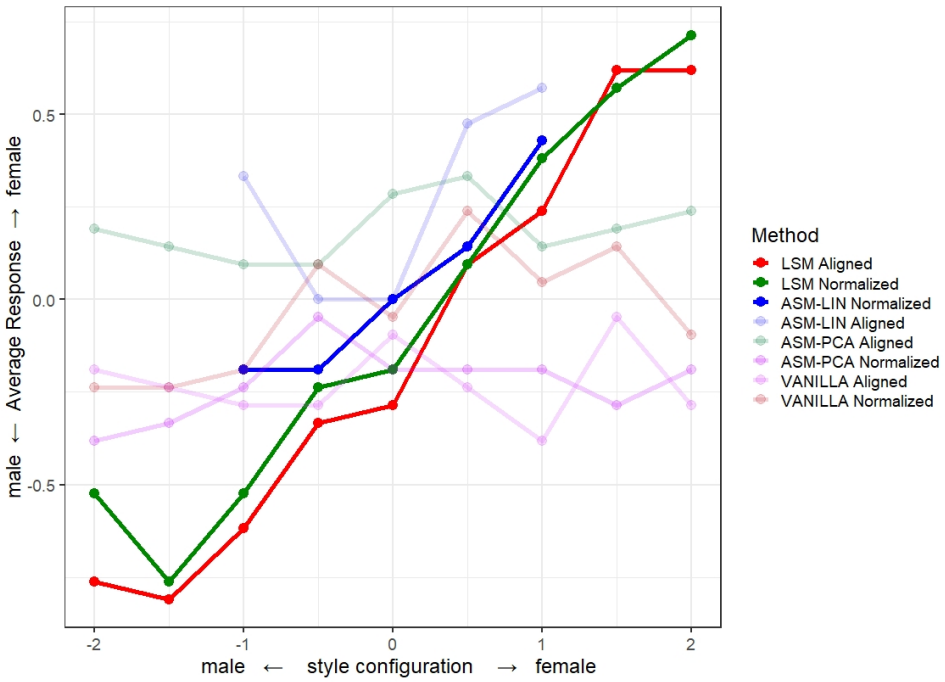
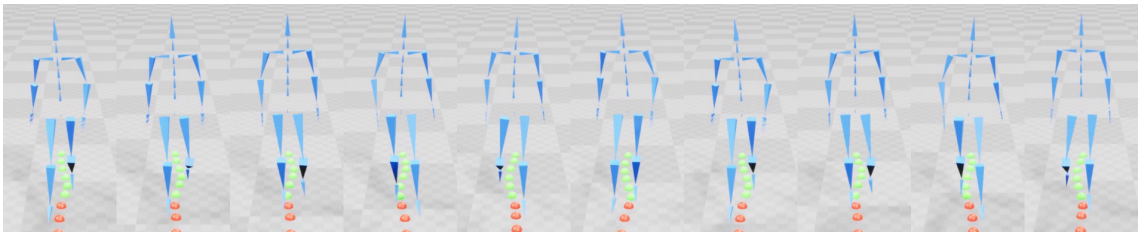
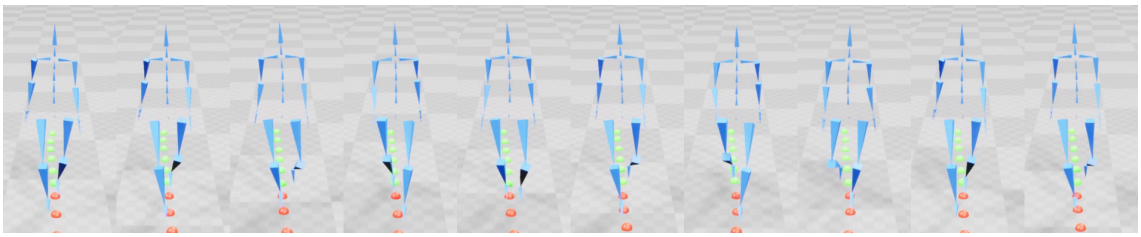


Figure 4.6: Average rating for different style configurations. Male rating response is encoded as -1; female rating is encoded as +1. The methods producing motion that is recognized above chance level are displayed in solid colors. Results for other methods are displayed for completeness. Absolute style values larger than one denote extrapolation of style. Reprinted with permission from [Spr+19b].



(a) Exemplary motion sequence for extrapolated male walking produced with the LSM aligned model. The motion displays pronounced shoulder movements and elbows pointing slightly outwards with only minor hip movements.



(b) Exemplary motion sequence for extrapolated female walking produced with the LSM aligned model. The motion displays a pronounced hip sway with only minor shoulder movements and elbows pointing slightly inwards.

Figure 4.7: Exemplary motion sequences for male and female walker models.

However, previous evaluations using point-light-displays report a response bias towards the male spectrum, providing a neutral walking stimulus [TS10]. Animations generated with LSM replicate this finding (see Figure 4.6). As this effect is not apparent in the ASM-LIN method, it further underlines the validity of our proposed method.

Archetypal style models (ASM) are phase-functioned neural networks tweaked for a specific higher-level style. This approach corresponds to traditional work for stylization, where a single artificially generated style is available (e.g., sexy walking). Only direct linear interpolation between two archetypal models was recognized above chance level by human raters. However, this approach was unstable under extrapolation, and the recognition performance was worse than that of LSMs. This result suggests that the traditional approach of a single style example is not comprehensive enough to replicate natural locomotion styles.

Troje [Tro02] pointed out that there is a dynamic and static component of motion. Male and female participants differ not only in their dynamic motion but also in their physiognomy. This static difference was approximated by the capturing device used in this work. The impact of static and dynamic input motion was evaluated using a normalized dataset with a single skeleton definition and an aligned dataset containing different root-bone-aligned skeleton definitions. As static differences between participants reduce the classification error in a perception experiment [Tro02], they were excluded in the human evaluation to enhance comparability between models. In addition, the avatar and its skeleton usually provide static differences in real applications.

Training the LSM model on aligned data produces a marginally more expressive result than training on normalized data. The additional static information most likely amplifies the style-relevant differences inside the latent representation, enabling a better linear decomposition. However, all other methods perform better when using normalized data. This is most important for the ASM-LIN, which was the only one to be able to produce visually classifiable motion when trained on normalized data. As the archetypal networks only use a single network per style, additional static differences between participants seem to inhibit learning a proper representation of this style in a single network. These results suggest that although the static differences between participants can help to generate stylistic motion models in the case of LSM, they do not for ASM. Static differences during fine-tuning of a single network seem to confuse the network. Hence, a single network should only be trained on a single static skeleton model rather than actor-specific skeleton models.

Limitations

Although assumed, there is no proof of correspondence between network weights. Fine-tuning seems to retain a weak correspondence that is sufficient for further latent space modeling. The binary scale of the answer format is not ideal and should be replaced with a better rating scale. Also, the synthesized networks can generate more complex motions, but only simple walking and running motions were evaluated to keep the number of stimuli in an acceptable bound.

Future Work

Based on the results of this work, multiple new directions for further research have opened up. Following the logic of Holden et al. [HKS17], it may be possible to include the style information directly during training. Increasing the phase function's complexity or implementing a secondary gating network [Zha+18] might teach style information directly during network training. The binary nature of the current style annotation can be combatted by an additional user study ranking the expressiveness of gender or by using automatic ranking functions [Par+21; Cha+19].

The general approach of linear regression was previously applied by Troje [Tro02] on a separate latent representation for the decomposition of additional styles, like weight or happiness. His findings have two implications for the results of our work. First, multi-dimensional style learning from data using the LSM method should be possible, given the appropriate data. During our work, a multi-dimensional LSM was implemented to decompose gender and weight. However, due to the correlation of gender and weight as well as the comparably small dataset, this method produced valid motions for a limited amount of configurations and was not general enough to be included. Further research and more motion capture data are required for this task.

As described by McDonnell et al. [McD+09], the body shape and motion influence the sex perception of virtual avatars. Our results suggest that our LSM model can change the perceived gender of the motion in a controlled way. Although the results by Troje [Tro02] and McDonnell et al. [McD+09] suggest that structural information or body shape of an avatar will only increase the distinctiveness of an avatar's gender, this still has to be evaluated. This includes the evaluation of motion mapped to an androgynous avatar and congruent (male motion, male body shape) and divergent (male motion, female body shape) mappings.

The presented approach uses the same principle of combining multiple walker-specific latent representations using linear regression in model space as presented by Troje [Tro02] but with a completely different latent representation. The neural network utilized in our work uses three fully connected layers. Recent publications continue using such a three-layer approach. Zhang et al. [Zha+18] proposed a network model using a gating network rather than a phase-function to interpolate network weights, with their motion model remaining structurally the same. Lee et al. [LLL18] use an LSTM network with a three-layer architecture and similar motion encoding. We expect the presented approach to generalize to these similar motion models. However, further research is required to verify this hypothesis.

Fine-tuning is used to keep a semantic correspondence between network weights. As the approach of fine-tuning used in our work is very basic, a large amount of data was required for each participant. Novel fine-tuning approaches, such as few-shot learning, have already been proven to work with the motion network architecture [Mas+18]. These approaches might reduce the motion capture data required for each actor to only a few steps. Future research is required to evaluate few-shot learning with linear style models. If the effort required for a motion-capturing experiment is reduced, incorporating more samples is feasible and could improve future LSMs.

Figure 4.8: Intra-invidual variation visible in five exemplary characters animated by our model with the same initialization and control input.

4.3 Learning and Simulating Variation in Motion

Human locomotion is composed of two characteristics: consistency and variation. It is quite often considered a cyclic process (e.g., [Tro02; HKS17; Zha+18]), as consecutive steps are similar while walking with the same velocity and direction. However, performing exactly the same motion for two consecutive steps is physically challenging and usually slightly different. It is not only important to accurately and fluently replicate the mean motion but also to generate the intrinsic variation of natural locomotion in animations. The lack of variation becomes apparent when the character is perceived for a long time with the same input (e.g., walking straight ahead) or in a group with characters using the same animation model. In this case, even a very natural locomotion clip will appear robotic and unnatural. In novel application fields, like a digital reality [Dah+19] simulating pedestrian agents for testing autonomous driving systems, the variation is not only preferred but highly required. Suppose an autonomous vehicle is trained or validated in a virtual scene using avatars with only a single animation clip. In that case, the systems will overfit or only be proven to work with this single animation. Hence, it is necessary to replicate the apparent variation, as depicted in Figure 4.8.

The recent publications for real-time motion synthesis (e.g., [LKL10; HKS17; Zha+18]) presented methods that generate motions of very high quality, but these approaches usually lack variability due to their deterministic nature. Given the same initialization and user input, all motions appear the same. Although there are already approaches using variational autoencoders, they lack either the ability to responsively control the motion synthesis [MHM18] or the fidelity of the generated motion is not comparable with the results of state-of-the-art models [Hab+17]. Mixture-of-expert (MoE) networks have shown remarkable performance in generating responsive natural animations of locomotion [HKS17; Zha+18; Sta+19]. These networks consist of a core motion network and an interpolation method. The core motion network is a fully connected neural network and performs the regression step from the current to the next frame. The interpolation method is used to generate the optimal network weights of the motion network for the current regression step.

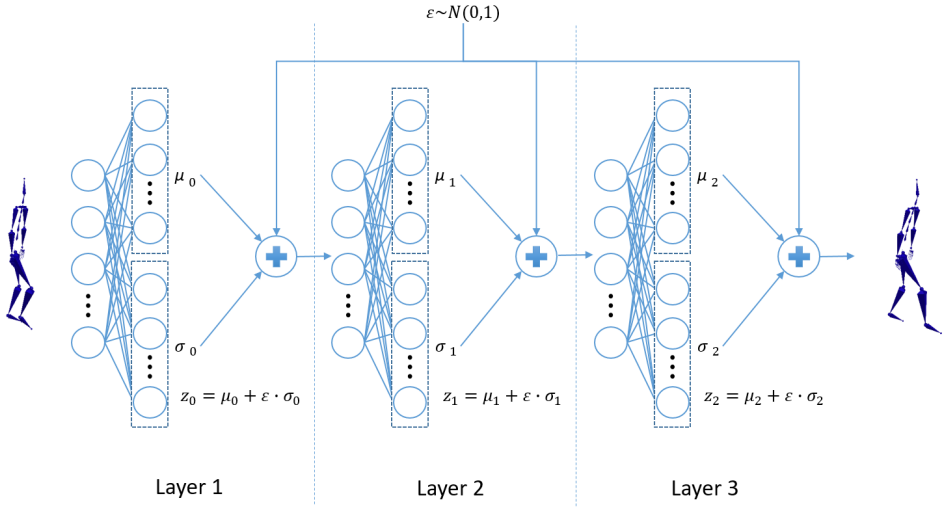


Figure 4.9: Network structure using three variational layers (vinn1n2n3). A single sample is drawn from a normal distribution and used to scale the trained distribution in the hidden layers. Reprinted with permission from [Spr+20].

This section presents a novel approach to adapt the core motion network of an MoE model in order to learn and reproduce the variation intrinsic to the training data while retaining the motion quality and responsiveness. Similar to a variational autoencoder, a hidden layer's parameters correspond to the mean and variance of a normal distribution. During model training and motion synthesis, a random variable is sampled from a normal distribution and scaled by the parameter. Hence, the model can learn the distribution inherent to the motion capture training dataset inside the hidden layer. The approach is based on a phase-functioned neural network (PFNN) [HKS17]. The impact of different numbers and positions of variational layers inside the network is evaluated against adding random noise and a vanilla PFNN. The targeted approach increases the inter- and intra-individual variation of locomotion while retaining the quality and naturalness of the original motion. This work has the following contributions:

- A novel approach to variational interpolating neural network (VINN) is presented.
- Evaluation of the impact of multiple hidden distributions in a single variational network.
- Novel insights for the usage of variational generative networks with a high frame rate.

4.3.1 Approach

By considering animation as a concatenation of individual poses, the whole process can be considered as a time-series problem where a regression model predicts each step. A separate imperative controller is required to maintain the avatar's global state and to enable a user to control the motion generation. Considering a fully connected or MoE network, a single latent representation performs the regression steps deterministically. We propose a novel approach for learning not a single representation but the distribution of the training data inside of a MoE model.

Neural Network Structure. For a single variational layer, the output of a fully connected layer is split into the mean (μ) and the standard deviation (σ) of a normal distribution. The output of the layer is computed by drawing a normal distributed variable $\epsilon \sim N(0,1)$ and scaling it by the values computed in the fully connected layer:

$$v(\mu, \sigma) = \mu + \epsilon \cdot \sigma \tag{4.3}$$

We can define this split function as $split(v) = (\mathbf{v_1}, \mathbf{v_2})$ with $\mathbf{v_1} = [v_1, v_2, ..., v_m]$ and $\mathbf{v_2} = [v_{m+1}, v_{m+2}, ..., v_n]$, where n is the length of the vector v and $m = \lfloor \frac{n}{2} \rfloor$. Using this split function, a single variational layer for an input vector x, weights W, and bias b can be defined as

$$\text{VAR}(W, b, x) = v(split(Wx + b)) \tag{4.4}$$

Using the exponential linear units (ELU) as an activation function, the variational layer can be written as:

$$\text{VAR}^{(e)}(W, b, x) = v(split(ELU(Wx + b))) \tag{4.5}$$

Given an input parameter $x \in \mathbb{R}^n$, output parameters $y \in \mathbb{R}^m$ and a phase parameter $p \in \mathbb{R}$, we construct a three-layer network as visualized in Figure 4.9. Mathematically, the network can be described as

$$\Phi(x; A) = \text{VAR}_3(W_2, b_2, \text{VAR}_2^{(e)}(W_1, b_1, \text{VAR}_1^{(e)}(W_0, b_0, x))) \tag{4.6}$$

A network model A can be completely defined by $A = \{W_0 \in \mathbb{R}^{2h \times n}, W_1 \in \mathbb{R}^{2h \times h}, W_2 \in \mathbb{R}^{m \times h}, b_0 \in \mathbb{R}^{2h}, b_1 \in \mathbb{R}^{2h}, b_2 \in \mathbb{R}^m\}$. Here, $h = 512$ is the size of the hidden state, $n = 301$ is the dimension of the input vector, and $m = 245$ is the dimension of the output vector. Similar to [HKS17], we employ a phase-functioned structure on this network. Hence, the network weights are computed by a phase function before each regression step, depending on the phase and the input parameters θ. In this case, a cyclic cubic Catmull-Rom spline (see Appendix A.1.3) is chosen as the phase function for control points $\theta = \{\alpha_0\ \alpha_1\ \alpha_2\ \alpha_3\}$.

Input and Output Format. The network performs a single regression step from the past to the future frame. The input and output vectors are defined as a combination of control input and joint configurations. The input vector contains trajectory positions $p_i \in \mathbb{R}^{2f}$, trajectory directions $d_i \in \mathbb{R}^{2f}$, and gait controls $g_i \in \mathbb{R}^{f \times s}$ of $f = 12$ surrounding frames and the $s = 6$ target gaits (standing, walking, jogging, etc.). The past joint configuration is provided as joint positions $l_{i-1} \in \mathbb{R}^{3j}$ and joint velocities $v_{i-1} \in \mathbb{R}^{3j}$ for all $j = 31$ joints.

The output vector contains global translation $r_i^x \in \mathbb{R}^2$, the change of forward direction $r_i^a \in \mathbb{R}^2$ projected onto the ground plane, the phase delta (elapsed phase) $\Delta_i \in \mathbb{R}$. In addition, it contains the predicted future trajectory positions $p_{i+1} \in \mathbb{R}^f$ and directions $d_i \in \mathbb{R}^f$. Last but not least, it contains the predicted next posture as joint positions $l_i \in \mathbb{R}^{3j}$ and velocities $v_i \in \mathbb{R}^{3j}$ for all $j = 31$ joints.

The full input and output vectors are thus:

$$\begin{aligned} x_i &= \{p_i \;\; d_i \;\; g_i \;\; l_{i-1} \;\; v_{i-1}\} \in \mathbb{R}^{306} \\ y_i &= \{r_i^x \;\; r_i^a \;\; \Delta_i \;\; p_{i+1} \;\; d_{i+1} \;\; l_i \;\; v_i\} \in \mathbb{R}^{215} \end{aligned} \tag{4.7}$$

4.3.2 Evaluation

Dataset. All frames from the motion capture library published by Holden et al. [HKS17] containing only the locomotion on a flat surface were utilized to reduce the variability introduced by external factors, such as the terrain. After pre-processing, the raw motion capture data in the required format, 179,586 frame-pairs, were extracted.

Network Training. The network architecture was implemented using the TensorFlow [Aba+15] framework. Each network was trained on the same dataset for 50 epochs using the stochastic gradient descent algorithm Adam [KB17], already included in the framework. Dropout was applied with a retention probability of 0.7. All models were trained in mini-batches of size 32. Cosine decay with warm restarts adjusted the learning rate (initial: 10^{-4}) over time, with restarts at 10 and 30 epochs.

Number of Variational Layers. We are utilizing a three-layer neural network. Hence, we can utilize up to three variational layers as described in Equation 4.6. Each variational layer, however, increases the complexity of the network. Hence, the effects of the number and the position of variational layers in the network were evaluated. We describe these networks using the notation *vinn1n2n3*, where the numbers describe the variational layers in the network and are left out if there is a non-variational layer at the specific position. For example, *vinn1n3* describes a network with a variational layer in the first and the third position and a regular (non-variational) layer in the second position.

Dimensionality of Random Sample. The random sample ϵ is drawn from a normal distribution and can be either of a dimensionality of one (single real number) or the same dimensionality of the hidden layers. We evaluated the effects of both configurations. In pre-tests, we found that using a random sample in the dimensionality of the hidden layers is beneficial for both, quality and variation, and thus utilized a large dimensional random sample for all experiments.

Measuring Variation. An experimental setup was utilized to compare the variation in locomotion generation. Using a vanilla PFNN, 10 seconds of straightforward walking were simulated to initialize a walk cycle. Based on this initialization, all models generated 100 separate walk cycles of four seconds or about five steps. The following three aspects were measured for the generated walk cycles. The **inter-individual variation**, comparing variation between the walk cycles for each time-step. For the baseline-method (PFNN) this variation should be zero. The **intra-individual variation**, describing variation throughout a single walk cycle. And the **error**, measuring violations of smoothness, foot drifting, and large inconsistencies between steps (stumbling).

The **inter-individual variation** (V_{inter}) was computed by the sum

$$V_{\text{inter}} = V_{\text{inter pose}} + V_{\text{inter step}} \tag{4.8}$$

The *inter-pose variation* ($V_{\text{inter pose}}$) was measured as the median deviation of local joint positions of the mean joint configuration for each time step. The *inter step velocity* ($V_{\text{inter step}}$) was measured as the average standard deviation of step velocity of temporally aligned steps generated by the different models. The sum of both variables was used to measure the inter-individual variation.

The **intra-individual variation** (V_{intra}) was computed as the sum

$$V_{\text{intra}} = V_{\text{intra pose}} + V_{\text{intra step}} \tag{4.9}$$

The *intra-pose variation* ($V_{\text{intra pose}}$) was measured by (i) grouping all generated local joint positions to their respective phase of the walk cycle (in 0.01 increments) and (ii) computing the average deviation of joint positions for each phase increment in the same way, as the inter-pose variation. The *intra step velocity* ($V_{\text{intra step}}$) was measured as the average deviation of step velocity of all steps generated from the specific motion model.

The **error** was measured as the sum

$$\text{error} = e_{\text{smooth}} + e_{\text{drift}} + e_{\text{distance}} + e_{\text{duration}} \tag{4.10}$$

Smoothness (e_{smooth}) was computed as the average of the 95-percentile velocities generated by each model. *Foot drifting* (e_{drift}) was computed for each generated walk cycle and averaged. *Step distance consistency* (e_{distance}) and *step duration consistency* (e_{duration}) are computed by taking the differences of distance and duration of consecutive steps. Consistency and smoothness should be reasonably small but not zero, as there would be no dynamic movement in this case.

Numerical Results. The VINN models are compared against a vanilla PFNN implementation and random models, where instead of sampling from a distribution, random noise is added to the layers of a PFNN since this could add variation as well. In addition, sampling at each frame and sampling at each step is compared. In Figure 4.11, the results are presented as scatter plots of the variation versus the error. The baseline model produces zero inter-individual variation and only minor intra-individual variation. All random models generate motion with high error scores but do not necessarily generate more variation. Sampling at each step rather than each frame can double the variation scores without increasing error in the VINN models. The best-performing model for inter- and intra-variation with an error comparable to the baseline model contains only variational layers (vinn1n2n3).

Qualitative Results. The different models were used to generate locomotion in different scenarios. Figure 4.10 displays the result for the target model (vinn1n2n3). The qualitative analysis supports the numerical evaluation. Adding random noise creates a high amount of jitter when sampling at each frame. When sampling at each step, fewer variations in motion but skating artifacts can be

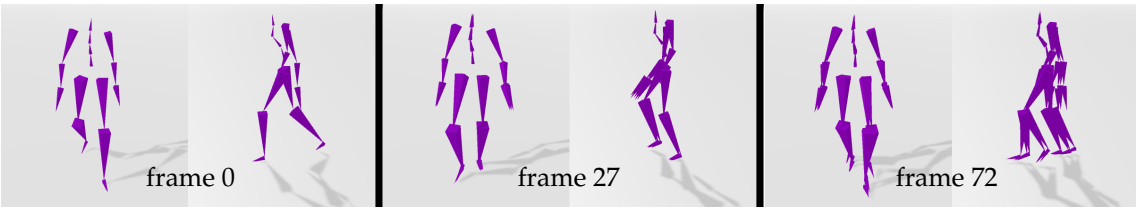


Figure 4.10: Development of motion variation within the first 72 frames (1.2 s). The display shows five overlayed skeleton walkers fixed at their hip position from front and side view. The motion is generated using single-step sampling with the vinn1n2n3 model. Reprinted with permission from [Spr+20].

observed, where the global translation does not match the body movements and the standing legs are not fixed on the ground. In the case of VINN, the visible variation increases for all models when sampling at each step. The body movement corresponds to the global translation very accurately. Although there is a small numeric discontinuity at the specific frame of the change of random sample, the discontinuity is barely visible in the animations.

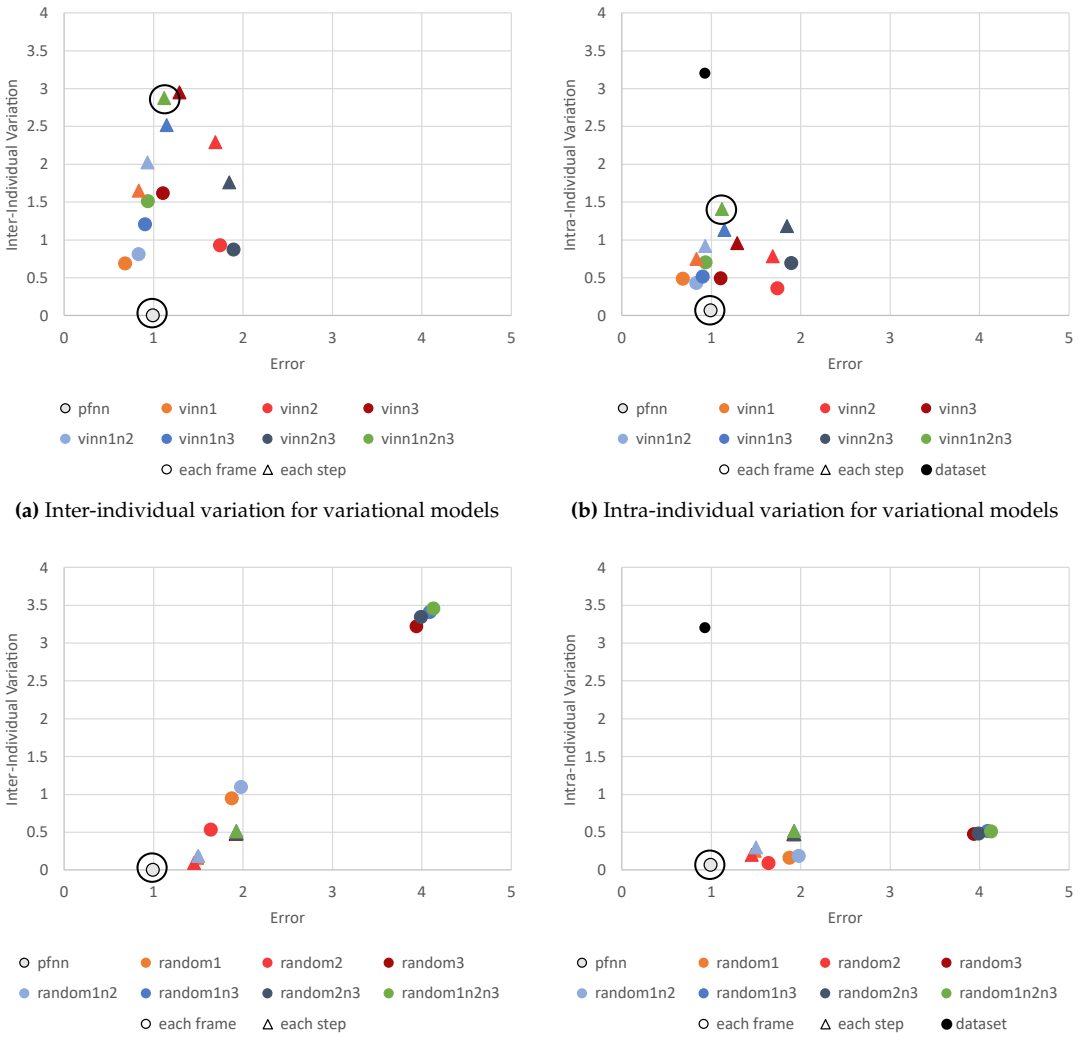
4.3.3 Discussion

The evaluation results suggest that the proposed network can capture and replicate the natural variation in motion at its best, if a distribution is trained in each layer. Compared to a vanilla PFNN, VINN generates a significant amount of variation. Compared to adding random noise at each layer, it generates consistent motions with respect to foot placement, step length, and smoothness, resulting in more natural animations.

The generated animations were compared against ground-truth samples of straight-walking which show a baseline error of 0.93 and a baseline intra-individual variation of 3.2. Although the variation is twice as large as generated by our model, it shows a comparable error. However, the ground-truth samples have small deviations from a straight trajectory, artificially increasing the variation. For future data capturing of locomotion, we recommend to include some samples of treadmill walking to capture a repetitive walk cycle and enable a better evaluation.

Due to the high frame rate (60 fps), the variation regresses to the mean if a random sample is drawn at each frame. Sampling at each step, however, creates natural motions with high variability. The discontinuity between the two steps is small and can be further reduced by smoothing the random sample. However, future work should investigate using more temporally stable methods that offer a smooth transition even with the change of the random prior.

The proposed approach demonstrates that concepts of VAE can be integrated into a MoE network structure while retaining the controllability and visual quality of the original approach. Hence, our approach has the potential to serve as a starting point for future experiments with similar network structures [Zha+18; Sta+19]. In addition, it builds a basis for future work in the direction of conditional variation and stylistic motion synthesis.



(a) Inter-individual variation for variational models

(b) Intra-individual variation for variational models

(c) Inter-individual variation for random models

(d) Intra-individual variation for random models

Figure 4.11: Scatterplots of variation versus error for all models split by inter- and intra-variation factors. The baseline and the best-performing model (vinn1n2n3n) are encircled. All random models generated inconsistent and noisy motion. The intra-individual variation and error within the training dataset are denoted in black. The inter-individual variation of the dataset was not comparable, as there were no samples starting at exactly the same initialization and following the same directional input. (cf. [Spr+20])

4.4 Multi-Action Control of Motion

Mixture-of-expert models generate impressive motion quality but are optimized for a single task like locomotion [Spr+19b], basketball [Sta+20], or mixed martial arts [Sta+21]. Within the task, the priority is set upon responsiveness to human user input, but not a precise control of body parts. For example, boxing is simulated without specifying a precise goal position of the hands [Sta+21], and locomotion is simulated with different gait types and avoiding obstacles naturally [Sta+19] but not walking to a precise locomotion goal. Especially for pedestrian simulation, this is an issue as missing the goal target and stepping on the street unplanned might result in an unintended crash situation.

For pedestrian simulation, the control of gaze to look for the traffic (Section 4.4.1), foot placement while stepping up and down the curb or ascending staircases (Section 4.4.2), and hand movements (Section 4.4.3) are essential, for example, to press the traffic light button. These actions differ in their correlation with the gait. The gaze direction and footstep placement are correlated with the gait and movement direction – we typically look into the direction of walking and place our feet in a regular pattern if we do not perform a secondary task. Thus, breaking this correlation, which is typically learned in a neural network, and controlling the gaze and the foot placement is non-trivial. Hand movements, on the other side, can be uncorrelated and executed asynchronously with the locomotion, bringing additional challenges.

We investigated methods to exert more accurate control over the generated motion for these actions. The results of these investigations are summarized in the next subsections. Unfortunately, the generalization of models is limited. The controllers require intensive implementation, and the models are highly sensitive with regard to the data utilized for training, the hyperparameters of data processing, procedural blending parameters, and neural network parameters. Although the results presented in this section are of limited quality and success, the approaches and insights gained during our experiments can be helpful for future generations of researchers.

4.4.1 Controlling Locomotion Correlated Movements on the Example of Gaze

For pedestrian simulation, the control of the gaze direction is one of the most important aspects after the control of the movement direction, velocity, and gait. The gaze direction is utilized as a criterion in multiple methods for predicting crossing behavior [RKT18; ZB23]. To control the gaze, we extend the motion simulation method presented in Section 4.3 with an autoregressive gaze trajectory and blending approach. In addition to the root trajectory, the head direction trajectory $h_i \in \mathbb{R}^{36}$ around a window of one second to the future and one second to the past is extracted for every data point. Similar to the root trajectory, it is subsampled to every tenth frame. The full head trajectory is added to the input vector, and the prediction of the future trajectory is added to the

(a) Wait **(b)** Look left **(c)** Look right **(d)** Cross

Figure 4.12: Controlling the gaze with an MoE network trained our captured data and a custom controller. The example shows an illegal and, thus, potentially critical crossing maneuver on a red light.

output vector. This extends the network in- and output vectors to:

$$\begin{aligned} x_i &= \{p_i \;\; d_i \;\; g_i \;\; l_{i-1} \;\; v_{i-1} \;\; h_i\} \in \mathbb{R}^{342} \\ y_i &= \{r_i^x \;\; r_i^a \;\; \Delta_i \;\; p_{i+1} \;\; d_{i+1} \;\; l_i \;\; v_i \;\; h_{i+1}\} \in \mathbb{R}^{233} \end{aligned} \tag{4.11}$$

The procedural motion controller is extended with a specific target gaze direction (e.g., provided by a gamepad controller, the mouse, or an agent behavior). The head trajectory is blended toward the gaze direction to adjust the auto-regressive input, similar to the root trajectory.

The motion capture data published in the past [HKS17] for the training of a game character contains a strong correlation between the gaze direction and the locomotion direction. Trained on this data, the control of the gaze direction is highly limited, and only a few gaze directions can be targeted accurately. Especially during locomotion, the character primarily looks in the direction of the movement, and the control of gaze direction has a minor influence on the animation. Trained on partial data captured in our experiments (Section 3.3), the performance improved, but still, controlling gaze directions not within the data distribution was only partially possible. We captured a small dataset targeted to increase the variability of the gaze directions. A single motion actor performed locomotion tasks (idle, walk, run, transition) while looking in the movement direction half of the time and half of the time looking in other directions as the movement direction. Trained on this data, the approach could effectively control the gaze direction in the whole range of motion. An example of the generated motion in a driving simulator is shown in Figure 4.12.

During the dedicated motion capture session, the motion actor received a phone call, and the corresponding telephone motion during idling (standing) was recorded. The data was intentionally kept in the training data. During simulation, when the gaze was controlled to the bottom left or right of the character (looking at the phone) and directly afterward to the front, the character lifted the hand towards the head, mimicking the "talking on the telephone" motion. The motion was maintained during idle until the gaze was directed to the floor again. When transitioning to walking, the hand pose was maintained for multiple steps.

This effect highlights not only the possibility of the approach but also its pitfalls. Given the correct training data, creating highly controllable and domain-specific motions is feasible. However, increasing the complexity and exerting control as opposed to the correlations found in the data is challenging. In addition, unintended motions that are not annotated and controlled can have unintended effects on motion generation. We further explored the ability to adjust the motion within the range of the training data distribution on the example of footstep placements, which are important to simulate the accurate stepping up and down of the curb.

4.4.2 Controlling Footstep Placement

Whenever a virtual pedestrian enters a road, it usually involves stepping down from the curb onto the road's pavement. Utilizing height maps as an additional input to the neural network can help to generate realistic movements on terrain [HKS17]. However, this approach leads to imperfect foot placement, especially on regular surfaces like stairs or a single step up and down the curb. For example, (i) if the foot is placed only partially on the higher surface, the foot clips into the geometry of the staircase, (ii) if the foot slides slightly to the front or the back, it can be recognized to be on a different height, resulting in rapid adjustments (jumping) of the foot, and (iii) if the foot is only touching the staircase in the area of the ball of the foot, it might appear to be floating in mid-air.

Existing research on footstep planning [BPE10; Bea+15] utilizes traditional search methods, optimization, and blending and outperforms neural networks in accuracy but not in visual quality and responsivity. In the following, I present work conducted together with Rolando Morales Suárez in his master thesis [Suá21], combining concepts of footstep planning with MoE-based animation models. We specifically focused on staircase walking, in which most methods fail to place the feet accurately on individual treads, resulting in gliding over the staircase rather than taking one tread with each step.

Method

Our approach extends the neural state machine (NSM) [Sta+21] to incorporate footstep information. Similar to the approaches outlined above, NSM is a mixture-of-experts-model. A visualization of the approach can be found in Figure 4.13.

For our approach, the following input sources are utilized:

- The frame input $F_i = \{j_i^p, j_i^r, j_i^v,\ \ t_i^p, t_i^d, t_i^a,\ \ f_i^{left}, f_i^{right}\}$ consisting of the character pose in root local space (joint position $j_i^p \in \mathbb{R}^{69}$, rotations $j_i^r \in \mathbb{R}^{138}$, velocities $j_i^v \in \mathbb{R}^{69}$), past and future root trajectory information (positions $t_i^p \in \mathbb{R}^{26}$, directions $t_i^d \in \mathbb{R}^{26}$ and gait $t_i^a \in \mathbb{R}^{52}$, with the four gaits idle, walk, ascend, descend), and the past and future foot trajectories (left foot $f_i^{left} \in \mathbb{R}^{39}$, right foot $f_i^{right} \in \mathbb{R}^{39}$). Both trajectories contain information of a two-second time window around the current frame subsampled at 10 frames.

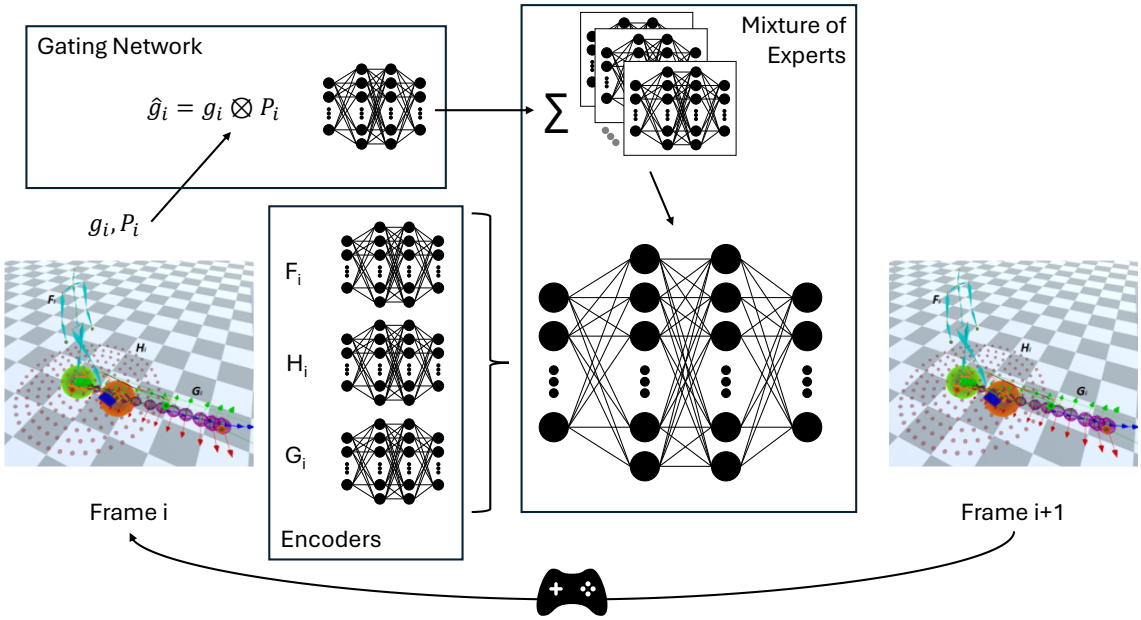


Figure 4.13: Concept of the neural state machine approach for staircase walking. The input is encoded with three encoder networks. The gating input g_i is combined with the locomotion phase P_i using the Kronecker product. The output of the gating network is used to combine 10 expert networks into a single mixture-of-expert model that is used to predict the next frame. An animation controller maintains the current state of the character, computes the environment information, and integrates the user control input (movement direction).

- The heightmap information $H_i \in \mathbb{R}^{528}$ around the character in the current frame.
- The goal input $G_i = \{g_i^p, g_i^d, g_i^a, \ g_i^{left}, g_i^{right}\}$ containing the goal trajectory one second in the future (positions $g_i^p \in \mathbb{R}^{39}$, directions $g_i^d \in \mathbb{R}^{26}$, and gait $g_i^a \in \mathbb{R}^{52}$) as well as the goal positions of the feet one second in the future (left foot $g_i^{left} \in \mathbb{R}^{39}$ right foot $g_i^{right} \in \mathbb{R}^{39}$. The trajectory contains information in a two-second time window around the target frame one second to the future subsampled at 10 frames.
- The gating network input $g_i = \{t_i^a, g_i^a, \ \delta \cdot t_i^a, \theta \cdot g_i^a\}$ containing the trajectory gaits t_i^a and goal gaits g_i^a as well as the distances to the goal position δ and the angular difference to the goal direction θ.

The inputs F_i, H_i, and G_i are each encoded to a latent space using three separate fully-connected neural networks with three layers and the ELU operator as an activation function. Each network has its own weights W and biases b and can be formulated as shown in Equation 4.12. The motion encoder F has a hidden layer and output layer size of 512, the heightmap encoder H of 512, and the goal encoder G of 128.

$$\Theta(X, W, b) = W_2 \times ELU(W_1 \times ELU(W_0 \times X + b_0) + b_1) + b_2 \tag{4.12}$$

The input to the gating network $\hat{g}_i = g_i \otimes P_i$ is the Kronecker product (see Appendix A.1.2) of the gating input g_i with the phase vector $P_i = \{sin(p_i), cos(p_i)\}$ where p_i is the current locomotion phase of the gait (see Section 4.2). The gating network is a fully connected three-layer network

(Equation 4.12) with a hidden layer size of 512. It predicts the blending coefficients w, which are used to combine the parameters α_i of 10 expert networks to the parameters of the motion network with a weighted sum $\alpha = \sum_{i=1}^{K=10} w_i \alpha_i$. The motion (and expert networks) are again a fully connected three-layer network (Equation 4.12) with a hidden layer size of 512.

The motion network is used to predict the next state of the frame based on the concatenation of the latent output variables of all three encoder networks. The output Y_i contains the concatenation of the prediction of the next state of the variables outlined above $\{j^p_{i+1}, j^r_{i+1}, j^v_{i+1},\ \ f^{left}_{i+1}, f^{right}_{i+1},\ t^p_{i+1}, t^d_{i+1}, t^a_{i+1},\ \ g^p_{i+1}, g^d_{i+1}, g^a_{i+1},\ \ g^{left}_{i+1}, g^{right}_{i+1}\}$ with additional variables $\{\tilde{j}^p_{i+1},\ \ \tilde{t}^p_{i+1}, \tilde{t}^d_{i+1},\ \ c_i, p_i\}$:

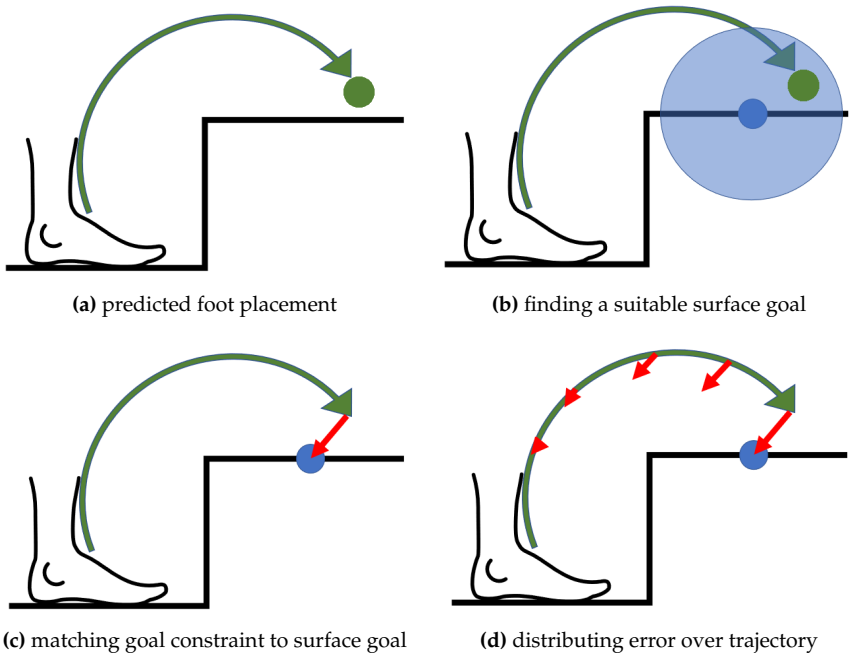
- The ego-centric joint positions $\tilde{j}^p_{i+1} \in \mathbb{R}^{69}$ one second in the future.
- The goal-centric trajectory positions $\tilde{t}^p_{i+1} \in \mathbb{R}^{12}$ and directions $\tilde{t}^d_{i+1} \in \mathbb{R}^{12}$ one second in the future.
- The foot contact labels $c_i \in \mathbb{R}^4$ denoting the contact of the feet and hands (not further utilized in this work).
- The update of the locomotion phase $p_i \in \mathbb{R}$.

The input data was normalized using z-normalization, and the network was trained using stochastic gradient descent with warm restarts (AdamWR) [LH19] with a weight decay rate of $2.5 \cdot 10^{-3}$ and an initial learning rate of 10^{-4} for a total of 150 epochs. Warm restarts were performed at epochs 11, 31, and 71. Training was performed in mini-batches of size 10, and dropout was utilized with a retention probability of 0.7. The model required around 4.5 hours of training on an NVIDIA RTX 2070 GPU.

In addition to the motion model, a procedural animation controller is utilized to process the input and output data, maintain the current state of the character model, and incorporate the user control (walking direction). The ascending and descending gait values are computed automatically based on the environment. Based on the prediction of the footstep trajectory, the position of the next footstep is estimated and checked with the environment. If the footstep is not on the ground or within an object, like the tread of a staircase, the future foot trajectory is adjusted to guide the foot placement on a free ground position. The adjusted footstep trajectory is fed back to the network for the next iteration. Figure 4.14 shows the concept of this adjustment.

Data

In a controlled capturing session, we captured the motion of an actor ascending and descending a longer staircase on the university campus. The Xsens Awinda system, an IMU-based capturing suit, was used to capture the motion data. The data was annotated and processed, and 22,813 frames at 60 FPS, or 6.3 minutes of clean motion capture data, were extracted. A virtual staircase was generated and fitted to the motion capture data. To increase the amount of variation in the training data, the virtual staircase was parameterized such that the regular tread width and riser height



(a) predicted foot placement **(b)** finding a suitable surface goal

(c) matching goal constraint to surface goal **(d)** distributing error over trajectory

Figure 4.14: Foot trajectory adjustment to reach a surface goal. (cf. [Suá21])

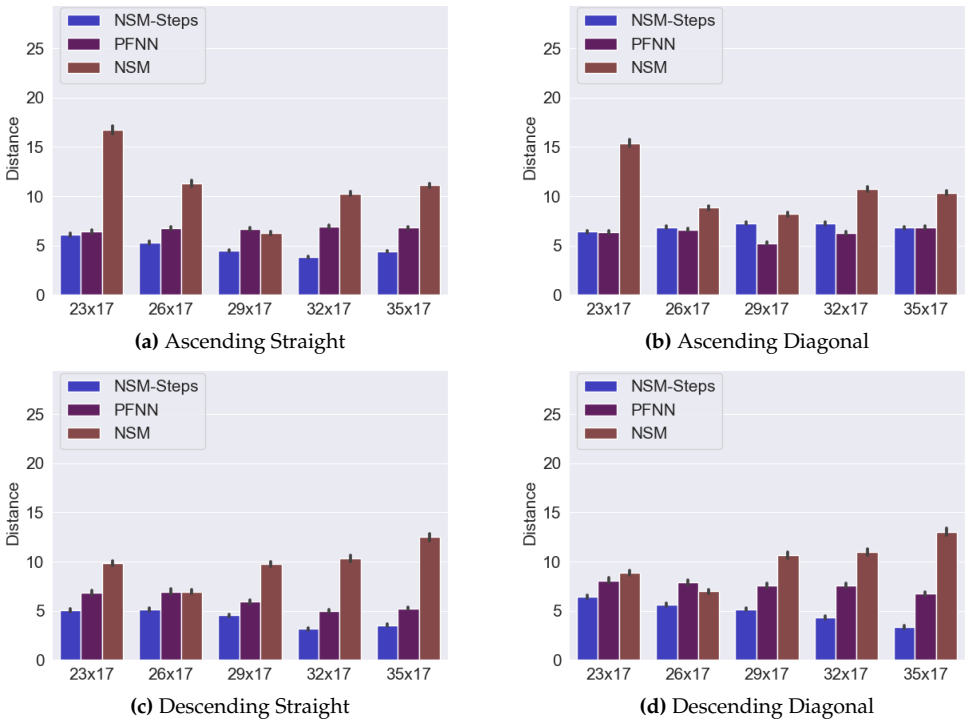
can be configured. Using an inverse kinematics solver, the existing motion capture data can be adjusted to match the new staircase configuration while remaining close to the original data. By mirroring the motion along the sagittal axis of the body, the amount of data was increased, and directional bias was reduced.

Evaluation

The extended **NSM-Steps** approach was compared against a standard multi-level neural state machine **NSM** [Sta+19] and a phase-functioned neural network **PFNN** [HKS17]. While **NSM** is the generally more performant model, it perceives the environment with a voxel map and is not specifically optimized for lateral movements. PFNN, on the other hand, perceives the ground via a height map and is developed for lateral movements on uneven terrain.

For the evaluation, the agents had to ascend and descend a 20-step stair 10 times with different tread depths and riser heights. Here, we focus on two measures to compare the different approaches: the distance of feet to the stair surface when the foot is placed on the stair and the number of steps required to ascend or descend the staircase.

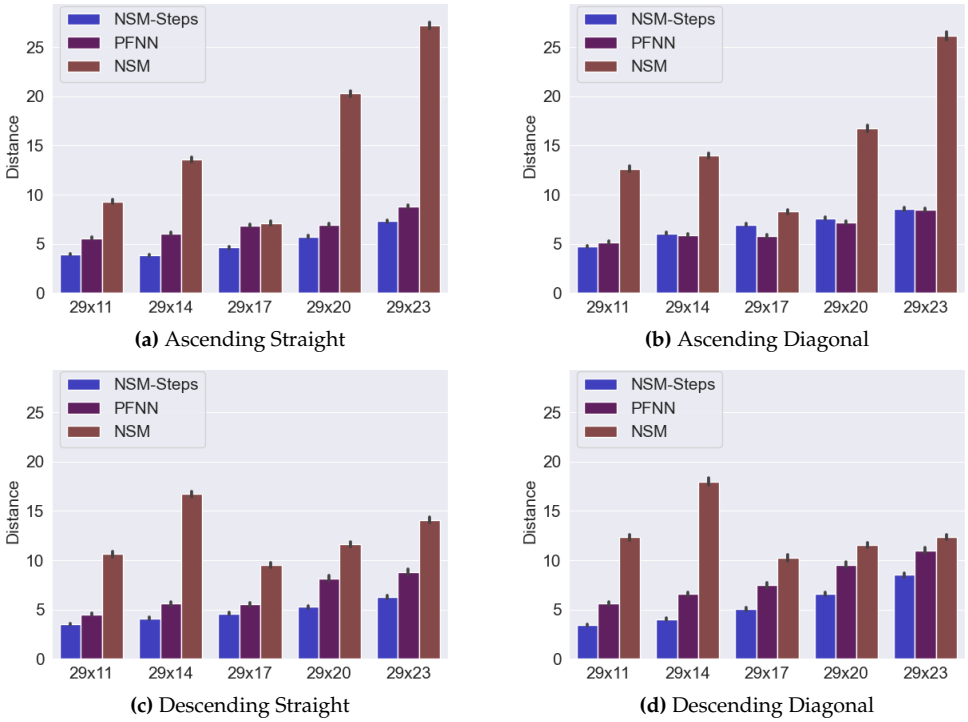
The quality of foot placement in the lateral dimension to the stair surface can be evaluated by measuring the distance of the feet to the ground. The lower the distance, the better the foot is placed on the stairs' surface. The sole height was measured at the center of the foot, in the middle between the toes and ankle. The absolute distance to the stair surface was utilized as an error measure. The results of the evaluation are shown in Figure 4.15 for different tread widths and in Figure 4.16 for different step heights.



(a) Ascending Straight **(b)** Ascending Diagonal

(c) Descending Straight **(d)** Descending Diagonal

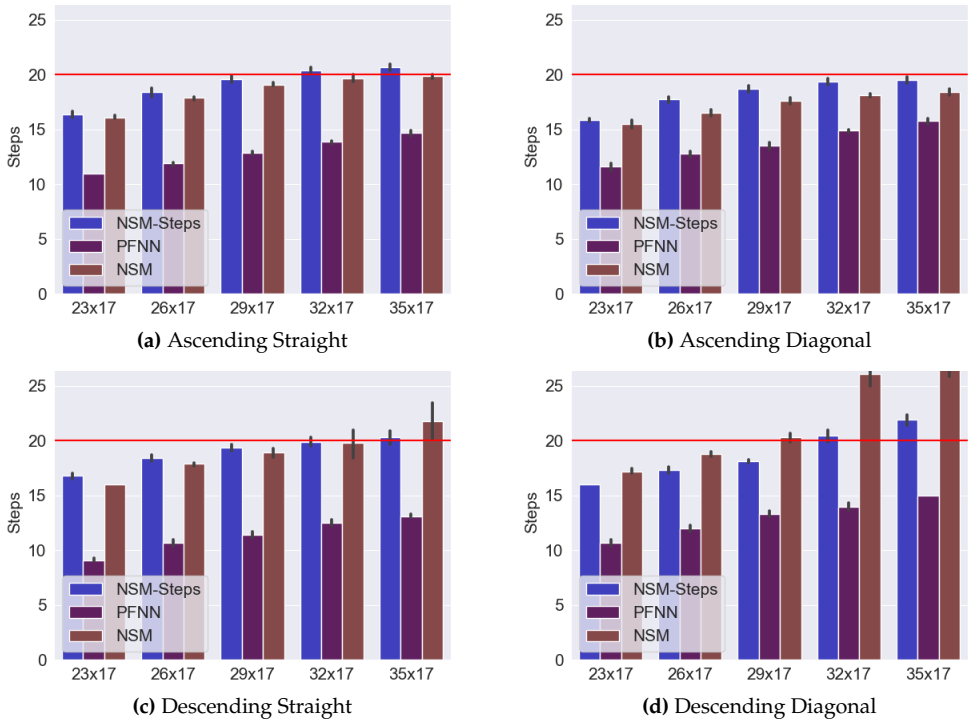
Figure 4.15: Distance of the ankle joints to the ground in cm for different tread widths. The ankle height above the ground of the actor is 5.3 cm. Error bars denote the 95% confidence interval. (Figures originally from [Suá21])



(a) Ascending Straight **(b)** Ascending Diagonal

(c) Descending Straight **(d)** Descending Diagonal

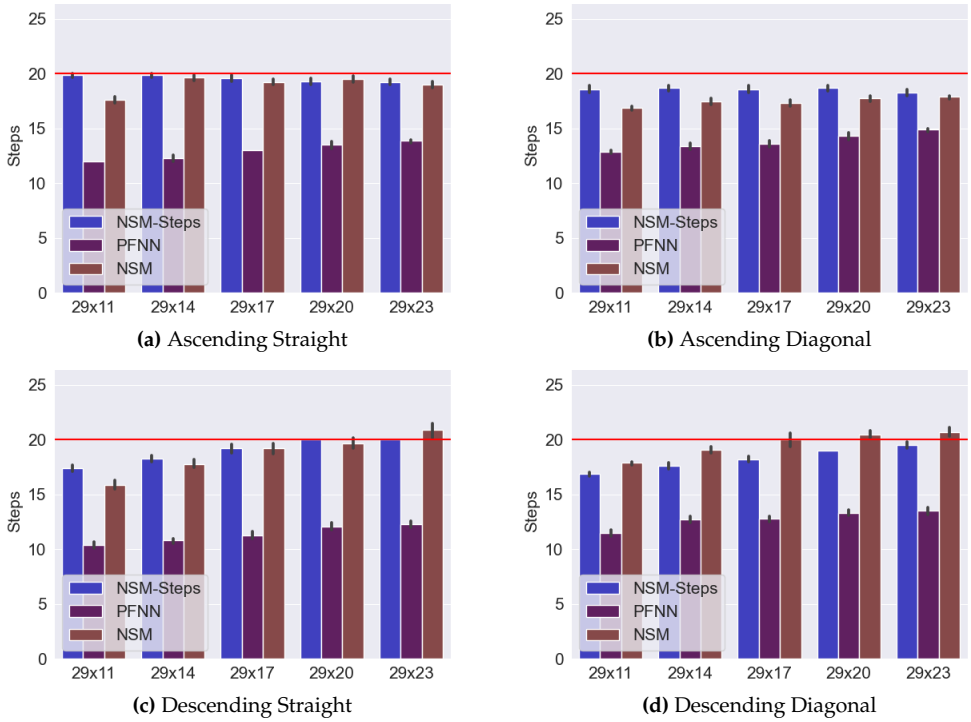
Figure 4.16: Distance of the ankle joints to the ground in cm for different riser heights. The ankle height above the ground of the actor is 5.3 cm. Error bars denote the 95% confidence interval. (Figures originally from [Suá21])



(a) Ascending Straight **(b)** Ascending Diagonal

(c) Descending Straight **(d)** Descending Diagonal

Figure 4.17: Number of steps required to pass a 20-step staircase for different tread widths. Error bars denote the 95% confidence interval. (Figures originally from [Suá21])



(a) Ascending Straight **(b)** Ascending Diagonal

(c) Descending Straight **(d)** Descending Diagonal

Figure 4.18: Number of steps required to pass a 20-step staircase for different riser heights. Error bars denote the 95% confidence interval. (Figures originally from [Suá21])

The **NSM** model shows the lowest performance, with the feet being consistently misplaced with a major distance to the ground. Especially when ascending, the model fails to place the feet appropriately. The **NSM-Steps** model outperforms the **PFNN** model for almost all stair configurations: The distance between ankle and staircase is about five cm for the **NSM-Steps** model (with an ankle height of 5.3 cm). The **PFNN** model generates animations with ankle distances of 5-25 cm, and this error consistently increases with increasing step height, as higher steps are more challenging to ascend and descend, as the foot must be lifted higher. With increasing tread width, the error decreases, as it is simpler to walk wider than very narrow steps.

For regular staircases, the number of footsteps to ascend and descend the stairs should be equal to the number of steps in the staircase. Therefore, the number of footsteps required to ascend and descend was calculated. Figure 4.17 shows the number of steps for different tread widths. Figure 4.18 shows the required steps for different step heights. All models fail to maintain a fixed number of steps to pass the staircase. In almost all cases, the models require fewer footsteps than the number of steps on the staircase. However, the **PFNN** model shows significantly more overstepping stair steps than the others, presumably because it was trained terrain walking with sloped surfaces allowing longer steps. As a result, the model is less affected by increased riser heights (see Figure 4.18). While the **NSM** and **NSM-Steps** model seems almost comparable in this measure, their difference becomes more apparent in extreme environments. The number of steps required to descend a staircase with 35 x 17 cm steps (width, height) in a diagonal direction results in a considerable increase in the required steps. While the **NSM-Steps** agent continues to descend the staircase consistently, the **NSM** model utilizes stairsteps multiple times (see Figure 4.17d).

Discussion

The presented approach significantly improves ascending and descending stairs by guiding the feet to the staircases. The subjective evaluation of the generated animations reflects the numeric results, with the **NSM-Steps** model producing the most plausible animations with the fewest errors. A further ablation study found that the foot trajectory had the biggest impact on the measured performance. While highly useful for pedestrian animation, the model does not allow for explicit foot control. It is impossible to control the character to kick someone or to place the foot in a specific location while waiting. In addition, the model cannot perform a stepping-stone task with a sequence of specific foot placements. One reason is most likely the strong correlation between the forward movement and the foot placement. Another reason is the implementation of the motion controller, which can only control the foot animation by interpolating the trajectories.

4.4.3 Controlling Fast Hand Movements

Although foot placement is most important for pedestrian locomotion, several actions require the control of the upper body limbs. While most pedestrian movements contain reaching motions to press a button or to open a car door, these actions are comparably slow. Boxing was selected

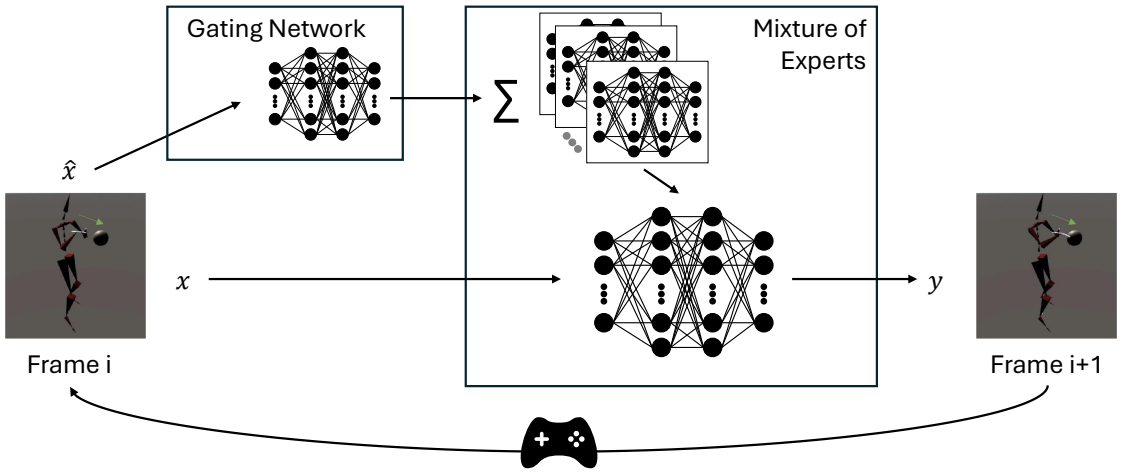


Figure 4.19: Concept of the mode adaptive neural network approach for boxing synthesis. The gating input $\hat{x}$ is passed to the gating network, which predicts the blending coefficients for eight expert networks. The experts are mixed with a weighted sum to create a frame-specific motion network that predicts the next frame. An animation controller maintains the current state of the character, computes the environment information, and integrates the user control input (movement direction).

as an application to test the utility of MoE models with indirect control of the upper body by interpolating joint trajectories. Fortunately, boxing rarely happens in real traffic situations, but boxing is a faster and thus more challenging hand movement compared to simply pressing a button. In the following, I will summarize the work conducted with Chirag Bhuvaneshwaran in his master thesis [Bhu21].

Method

A mode-adaptive neural network [Zha+18] is adapted to incorporate the wrist trajectories as a control input. It is a classical mixture-of-expert model using a three-layer gating network to predict the blending coefficients for eight expert models. The network structure of the gating and expert networks follows Equation 4.12. The gating network has a hidden layer size of 64, while the expert networks have a hidden layer size of 512. The gating network utilizes a subset $\hat{x}_i$ of the motion input x_i, outlined in the next paragraph.

The motion input $x_i = \{j_i^p, j_i^v \;\; t_i^{rp}, t_i^{rv}, t_i^{rd}, \;\; t_i^{wp}, t_i^{wv}, \;\; a_i^{pl}, a_i^{pt}\}$ consists of:

- The character pose in root local space in terms of the joint positions $j_i^p \in \mathbb{R}^{j\times 3}$ and velocities $j_i^v \in \mathbb{R}^{j\times 3}$ for all $j = 21$ joints.
- The root trajectory local to the current root position in terms of trajectory positions $t_i^{rp} \in \mathbb{R}^{2\times w}$, trajectory velocities $t_i^{rv} \in \mathbb{R}^{2\times w}$, and trajectory directions $t_i^{rd} \in \mathbb{R}^{2\times w}$ with $w = 10$ surrounding frames sampled at every tenth frame spanning from 0.83 s into the past and 0.83 s into the future

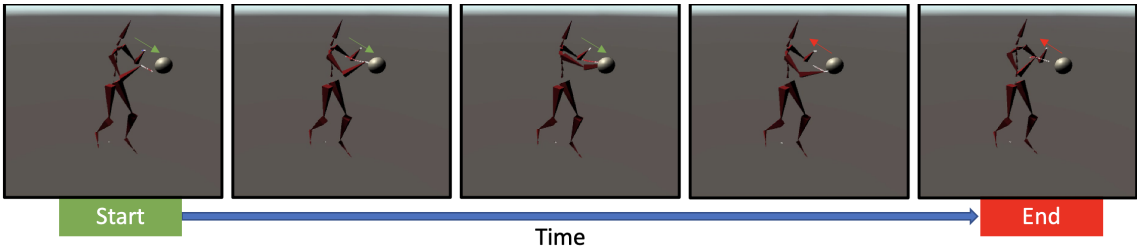


Figure 4.20: A punching sequence generated with our approach showing a typical animation for punching a target (gray). The trajectory is shown with small white dots and the punching label with a green arrow (punching) and a red arrow (retracting). (Figure originally from [Bhu21])

- The wrist trajectory local to the current root position in terms of the wrist trajectory positions $t_i^{wp} \in \mathbb{R}^{3\times v}$, and wrist trajectory velocities $t_i^{wv} \in \mathbb{R}^{3\times v}$ where v defines the number of surrounding frames (temporal window)
- The punch annotation $a_i^{pl} \in \mathbb{R}^2$ denoting whether a punch is executed by the left or right hand and the corresponding punch targets $a_i^{pt} \in \mathbb{R}^6$.

The trajectory lengths and the frequency of subsampling of the trajectories are varied in the evaluation detailed below. The best results are achieved with a trajectory length of $v = 14$ sampled every third frame.

The gating input $\hat{x}_i = \{t_i^{rp}, t_i^{rd}, t_i^{rv},\ \ t_i^{wv},\ \ a_i^{pl},\ \ w_i^v, f_i^p, f_i^v\}$ which contains a subset of the motion input in addition to:

- The velocity of the wrist end-effector joints $w_i^v \in \mathbb{R}^6$
- The position $f_i^p \in \mathbb{R}^{12}$ and velocity $f_i^v \in \mathbb{R}^{12}$ of the ankle and toe joints.

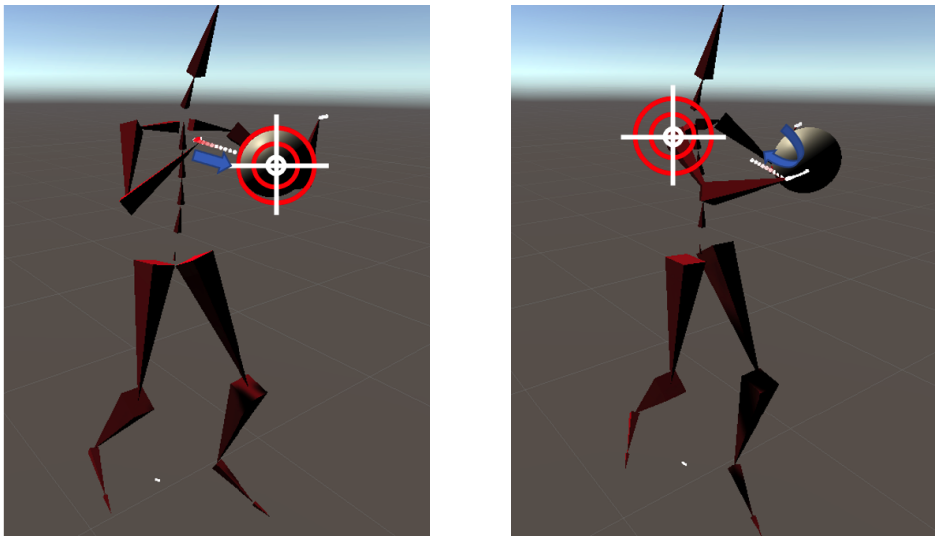
The output $y_i = \{j_{i+1}^p, j_{i+1}^v\ \ t_{i+1}^{rp}, t_{i+1}^{rv}, t_{i+1}^{rd},\ \ t_{i+1}^{wp}, t_{i+1}^{wv},\ \ a_{i+1}^{pl}, a_{i+1}^{pt},\ \ r_{i+1}^v, r_{i+1}^d, c_{i+1}\}$ of the motion network contains updates to the corresponding input variables. Note, that all trajectory output contains only the future trajectory points and thus has half of the dimensionality of its input counterparts. In addition, the output predicts:

- The change in root position $r_{i+1}^v \in \mathbb{R}^2$ and root direction $r_{i+1}^d \in \mathbb{R}^2$
- The foot contact labels $c_{i+1} \in \mathbb{R}^4$

Unlike our approach outlined in Section 4.4.2, the wrist trajectory is not synchronous with the root trajectory but utilizes a different sampling rate and length.

The model is trained with stochastic gradient descent with warm restarts (AdamWR) [LH19] with an initial learning rate of 10^{-3}, learning rate decay of $4 \cdot 10^{-3}$ and warm restarts after 10, 30, and 70 epochs. It is trained for 150 epochs in mini-batches of size 32, and dropout is applied with a retention probability of 0.8. Training required 30 to 120 min on an NVIDIA Quadro RTX 8000 GPU.

A custom animation controller was developed to control the boxing animations (upper body). It switches the punch action label and interpolates the wrist trajectory toward the target based



(a) Punching phase towards the target

(b) Retraction phase back to an idle stance

Figure 4.21: Visualization of the punching and retraction phase. Whenever the user triggers a punch, the punch controller interpolates the wrist trajectory of the respective hand toward the goal. Whenever the goal is reached, punching is disengaged, or the maximum time limit (25 frames) is reached, the controller interpolates the wrist trajectory back toward the idle position. (Figure originally from [Bhu21])

on human user input. Whenever the target is reached or the punch disengages, the trajectory is interpolated back to an idle position. If the system misses hitting the target, the animation controller must disengage the punching action to avoid an unnatural pose in which the character extends the arm unnecessarily long in a static pose. Thus, based on the statistics of the average punch duration, the punching phase is limited to a maximum of 25 frames. Figure 4.21 visualizes the approach, and Figure 4.20 displays keyframes from an animation sequence generated by the approach.

Data

The data of a single actor with basic boxing experience was captured using an Xsens Awinda motion-capturing system. The data contains around 10 minutes of motion data, including boxing with the leading hand (jab), the opposite hand (cross), punches coming from the outside to the inside (hook), and punches coming targeted from a lower to a higher point (uppercut). In addition, the data contains movement in different directions with typical boxing steps. The data is first processed with the Xsens software to reconstruct the global movement and then semi-automatically annotated in Blender to annotate the start and end of the punches. Most recorded punches have a duration of 25 frames at 60 frames per second and range between three (no full arm extension) and 50 frames.

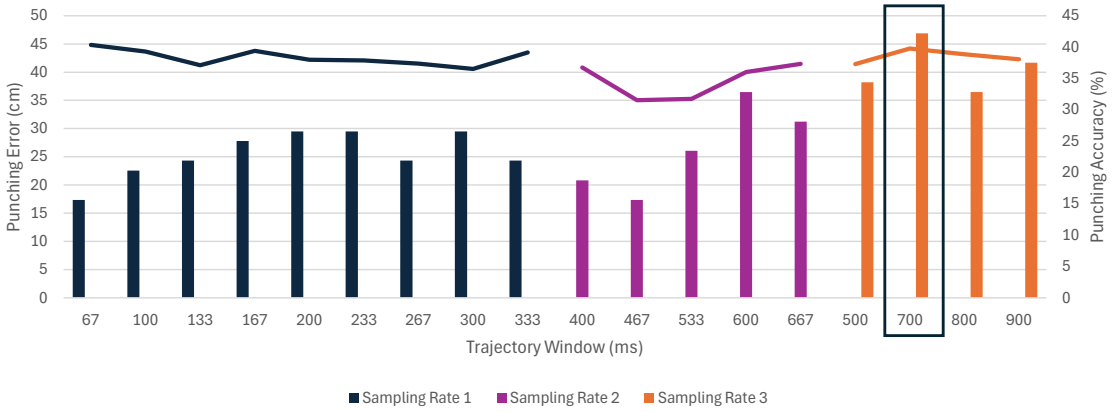


Figure 4.22: Average punching error and accuracy for different models using different trajectory window lengths and sampling rates. The graphs show the punching error, while the bars indicate the punching accuracy. The trajectory window is calculated by the number of trajectory points and the sampling rate (every first, second, or third frame). The punching error is averaged over all targets and both hands. A punch is accurate if it lands within 15 cm of the target location.

Evaluation

The trajectory length and the punch trajectory sampling rate vary between a trajectory length of 4 to 20 frames and a sampling rate between one to three frames. The resulting trajectory windows span 67 to 900 ms around the current frame. A single network is trained for the trajectory configuration. To limit the number of models trained, a subset of all configurations covering the whole duration was selected.

To evaluate the punching accuracy, 64 punch targets are uniformly sampled on a lattice spanning the range of motion in the captured data for each hand. The punch error is measured as the average Euclidean distance between the hand and the punch target at the frame in which the hand starts to retract. A punch is considered accurate if the distance is below 15 cm at that frame. The evaluation results are shown in Figure 4.22. The models trained with a trajectory that does not utilize subsampling but uses every frame of the trajectory (sampling rate 1) perform worst, having the highest punching error and the overall poorest punching accuracy. Comparing models trained with trajectories using sampling every second or third frame, a vital pattern can be observed. The models trained with the smaller sampling rate have a smaller error, while those trained with a larger sampling rate have a higher accuracy. As the punching error is computed as the average, it is sensitive to outliers. Hence, the model using 14 trajectory points and a sampling of 3 (700 ms trajectory window) is the most accurate model. Still, if it misses, it will miss with a higher error than the model with 16 trajectory points and a sampling of 2 (533 ms). Further ablation tests indicate a general decrease in performance when wrist joint trajectories and interpolation are removed and only an action label remains.

Visual inspection shows a high correlation between upper and lower body simulation. For example, if there is a step forward, the arms always move slightly in a punching motion, and vice-versa, whenever a punch is executed, the agent steps slightly forward. As this correlated behavior was

primarily observed in the data and thus is very natural, it is not desired from a control standpoint. While the control input seems to have some effect on the accuracy, the motion prediction depends primarily on the past pose and the action label.

4.4.4 Discussion

In this section, we presented three auto-regressive MoE models, which incorporate a more explicit control over body movements. In the first approach, gait-correlated actions are simulated, controlling the gaze direction using the head trajectory in the controller and network input. A second approach investigates foot placement using joint trajectory information in the controller and network input. In a third approach, hand movements using joint trajectory information of the wrists in the controller and network input are generated, and the effects of trajectory length and sampling were investigated.

All three approaches highlighted the importance of data for multi-action control. The intrinsic correlation between poses and action control makes arbitrary movements challenging, especially when moving individual limbs separately. Either the variation within the dataset needs to be maximized to provide sufficient samples for training, or a more elaborate system needs to be investigated in the future that can overcome the intrinsic data correlation. Both graph-neural networks [Sof+21] and hierarchical autoencoders [Li+19] seem to be good candidates for this task.

Several new approaches were recently published using auto-regressive MoE models of a similar style (e.g., [Sta+21; Sta+23]). Their visual performance is almost production-ready. However, they require highly curated datasets and domain-specific models for individual tasks like boxing and specific animation controllers to generate other animations than locomotion, which, in turn, require intensive implementation effort. Even so, the precise motion control reaching targets on a sub-centimeter scale remains a challenge.

While the overall performance of our approaches is not sufficient for publication, the insights gathered during the implementation and experiments were crucial for the decision to stop further investigations in this direction, given the available resources during my dissertation. Smarter approaches for controlling motion are required to reduce the amount of brute-force coding while achieving more precise animations. The procedural controller could be replaced with another neural network providing the control input to the motion network. Trained with reinforcement learning, a motion network could achieve a more accurate control with reduced implementation effort. In the next section, novel work in this direction is presented.

4.5 Multiple Agent Avoidance

Navigating virtual characters through a crowd is a long-lasting and well-established research area (we refer to [Yan+20; Kwi+22] for reviews). However, navigation systems rely on an abstract representation of a human character, like a capsule collider (two half spheres joined by a cylinder), which is either too small, resulting in geometry collisions, or too large, resulting in unnatural gaps between characters. The creation of natural avoidance movements using conventional methods (e.g., finite state machines, blend-graphs), with upper body rotations and small deviations of the navigation trajectory, is costly and often not within the budget for generic non-player characters (NPCs) [Hol18]. In non-gaming contexts, like crowd simulation for evacuation models, architectural visualizations, and traffic simulation, animations are usually not the primary target, resulting in limited animation data containing only the default walk cycle without any other actions at all (see [Ric20] for an in-depth comparison of simulation systems). A motion simulation model that can avoid a variable number of 'opponents' is required to simulate animations for these application areas. In addition, capturing the motion data of more people increases the requirements on the capturing equipment, annotation effort, and capturing effort beyond a reasonable amount for the targeted task. Thus, a suitable approach should be able to train on single-person capture data to reduce the capturing effort.

Data-driven approaches utilize motion capture data to create a generative motion model synthesizing new animations in a given environment. Several recent approaches have considered interactions between characters (e.g., [LLL18; Sta+21]) with incredible performance and plausibility, especially for locomotion tasks. However, most interaction simulations are primarily focused on two single individuals (e.g., [Sta+20; Men+22; Gho+23]) and require paired data-sequences for the proper simulation of interaction (e.g., [Fie+20]). Such an approach is not practical for crowd simulation, where multiple people are interacting with each other. Approaches considering more than one opponent are rare and usually restricted to a fixed number of opponents (e.g., [Wan+24]) and still require multi-person capture data. Physics-based motion models utilize optimization (e.g., [KLV20]) or reinforcement learning (RL)-based control policies (e.g., [Pen+17]) to control a physical representation of the human body. As a result, the model can react realistically to external perturbations, like bumping into someone else. While these models are great for simulating the reaction after an avoidance movement is unsuccessful, they do not automatically solve preemptive movements to avoid a collision in the first place. In addition, their computational complexity makes these approaches impracticable for NPC simulation.

Simulation models like adversarial skill embeddings (ASE) [Pen+22] and motion VAEs [Lin+20] combine a general-purpose skill network – trained to generate animation poses – with a high-level action policy providing the control input to the skill network via a latent code. Thus, the control policies, which are trained with reinforcement learning, exert the full body control indirectly via the skill networks. However, multi-person interaction has not yet been integrated into these types of models.

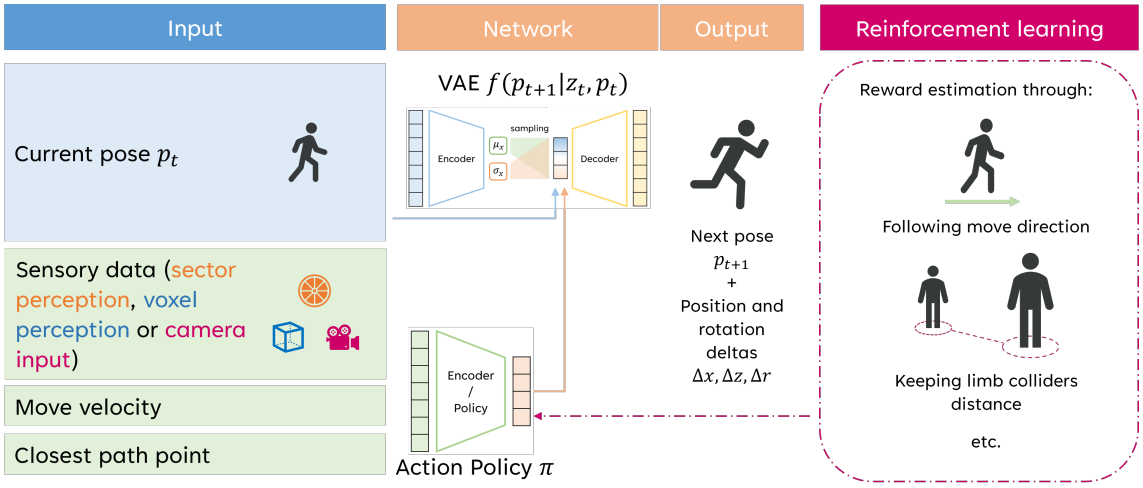


Figure 4.23: Complete architecture of the agent model architecture. The motion decoder is trained with motion capture data of a single actor. The action policy uses the perception systems to learn walking and avoiding skills through reinforcement learning and different rewards. (Figure originally from [Akm24])

In this work – based on the master thesis of Ulan Akmatbekov [Akm24] – we extend a motion VAE [Lin+20] to generate avoidance movements while following a navigation control input. The model requires only the motion capture data of a single person showing avoidance movements and can generalize to different number of "opponents". Different input types to perceive the characters are evaluated with respect to their performance, practicality, and computational cost. Our target model can generate plausible evasion movements when avoiding other agents in the scene with a low computational cost and no adjustments to the navigation system required.

4.5.1 Architecture

The model architecture comprises two neural networks: A motion model (skill network) and a control policy (action policy). The motion network is constructed as an autoregressive conditional variational autoencoder, predicting the next pose based on the past pose and a latent control input. The action policy generates a control input based on the environment observation and navigational control input. Figure 4.23 shows a visualization of the architecture.

Motion Generation

Pose Encoding. A single pose is encoded in root local space, with the root being the hip position projected to a flat ground plane oriented towards the agents facing direction. The root positional and rotational displacement is denoted with ($\dot{r}^x, \dot{r}^z, \dot{r}^a \in \mathbb{R}$). The joint rotations are represented by their forward and upward vectors ($j^r \in \mathbb{R}^{6\times21}$) and accompanied by the joint positions ($j^p \in \mathbb{R}^{3\times21}$) and joint velocities ($j^v \in \mathbb{R}^{3\times21}$). A single pose x_t at a given frame t is encoded as:

$$x_t = (\dot{r}^x, \dot{r}^z, \dot{r}^a, j^p, j^v, j^r)_t$$

Motion Model. The motion model is trained as a conditional variational autoencoder (cVAE). The encoder network Φ compresses the pose of the next frame x_{t+1} with respect to the current pose x_t to a latent motion manifold z_t:

$$\Phi(z_t|x_{t+1}, x_t)$$

The decoder network Ω is formulated as the conditional probability distribution of a character pose at the next frame x_{t+1} given the current pose x_t and the latent control variable z_t:

$$\Omega(x_{t+1}|z_t, x_t)$$

Action policy. An action policy π is trained using principle policy optimization (PPO) in a reinforcement learning (RL) setup to control the motion generation via the latent control vector z_t based on an observation s_t. The policy parameters are optimized to increase the reward for actions generated on a given state of the environment:

$$\pi(z_t|s_t)$$

Virtual Observation Sensors

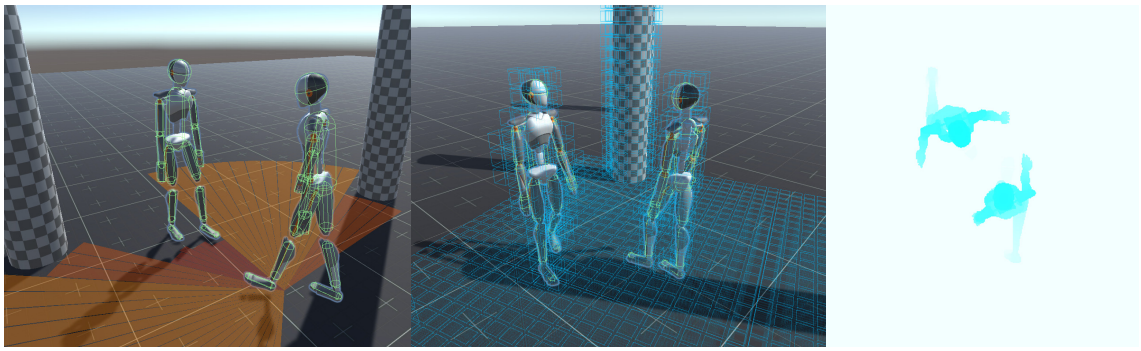
The observation contains the control input (directional velocity, closest path point, contact flag) in addition to the perception of other agents. Three different perception models are evaluated in this work. Example visualizations of the perception modules are shown in Figure 4.24.

Sector perception divides the area in front of the agent into discrete sectors on a circular plane. Each sector represents the closest distance to a dynamic or static obstacle projected to the ground plane. The field of view (FOV) is rotated with the movement of the agent and covers 220° as an approximation to the human FOV, and the distance is limited to three meters. The vector $S_t \in \mathbb{R}^N$ encodes the distance to obstacles for each of the $N = 44$ sectors bounded to a distance of three meters.

Voxel perception provides a volumetric perception in a cuboid volume around the agent, which denotes the presence of a physical object within its bounds at a certain frame t. The voxel space for a given number of voxels on each axis $(N_x, N_y, N_z) = (16, 8, 16)$ with a density of two voxels per meter can be defined as $V_t \in \{0, 1\}^{N_x \times N_y \times N_z}$.

Depth camera perception provides environment vision with a top-down depth camera observing the surroundings of the agent. Unlike a head-mounted camera, a top-down camera provides additional information in areas not directly perceivable by the agent. The depth camera perception vector is defined as $V_t \in \mathbb{R}^{32 \times 32}$.

Other perceptions have been evaluated but were considered insufficient for the target application. A single front-facing camera from the head perspective and a LIDAR-like sensor using ray-casting did not provide enough contextual input for a stable simulation. Multi-camera and panoramic cameras were computationally expensive and thus unsuitable for NPC simulation.



(a) Sector Perception **(b)** Voxel Perception **(c)** Depth Camera Perception

Figure 4.24: A visualization of the different observation sensors utilized in this work. Reprinted with permission from [Akm+25] © 2025 IEEE.

4.5.2 Data Acquisition

Motion capture data was recorded using an Xsens Awinda full-body system with 17 wireless inertial measurement units (IMUs) in a 3.5 x 7 m capturing area and two motion actors from the research group. In the first session, the locomotion of only one actor was captured to gather locomotion data in the absence of other agents. For this session, the actor walked to arbitrary points in the area, randomly stopping, idling, and resuming to walk. The velocity was adapted within a typical walking pace of 1.4 - 1.9 m/s and contained sequences of acceleration and slowing down.

The room to capture avoidance movements was augmented with two traffic cones. Two actors were asked to walk around the cones in an infinity symbol, resulting in straight-line segments and right and left turns. In addition, it encouraged the subjects to encounter each other more often, creating more avoidance movements. The actors were asked to (i) randomly stop, idle, and resume the motion, (ii) accelerate and decelerate randomly within walking speed (1.4 - 1.9 m/s), (iii) wait for the opponent in front and not bump into him, (iv) randomly perform spot-turns and switch the walking direction, (v) avoid collisions with the opponents as close as possible, involving shoulder movements. A visualization of this scenario can be found in Figure 4.25. Although two capture suits were utilized to investigate and visualize the movements, only the data of one actor was used for the training.

The raw motion capture software was first processed with the MVN Awinda software of Xsens to reduce sensor drift, foot skating, and magnetization. The resulting BVH animation data was further cleaned of any artifacts and additionally mirrored along the sagittal body plane to reduce motion bias and increase the amount of training data. The motion data was encoded into the pose encoding outlined above (see Section 4.5.1) and exported as training data sequences. A total of 38,116 frames at 60 frames per second or 10.6 minutes of motion capture data was utilized for training.

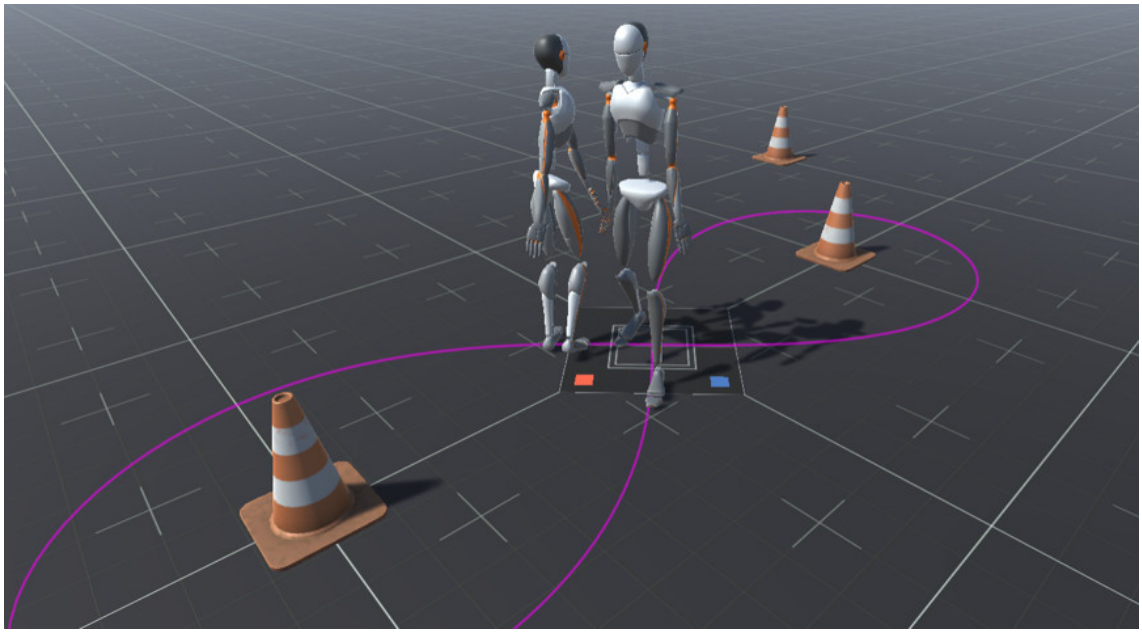


Figure 4.25: Example data of the avoidance capturing. Virtual cones and trajectories visualize the actual positioning of cones and target trajectories in the real capturing environment. Reprinted with permission from [Akm+25] © 2025 IEEE.

4.5.3 Implementation

Motion Model

The motion model is implemented in PyTorch [Pas+19] using the implementation of Ling et al. [Lin+20] as a basis. Figure 4.26 shows the detailed structure of the motion VAE network. The encoder Φ is implemented as a three-layer fully connected network with a hidden layer size of 512 and exponential linear units (ELU) as an activation function. Such a network is defined by its weights W_i and biases b_i as

$$\Phi(x) = W_3 \cdot \mathrm{ELU}(W_2 \cdot \mathrm{ELU}(W_1 \cdot x + b_1) + b_2) + b_3$$

The last layer is split to the mean μ and the squared logarithmic variance σ of the latent distribution, and thus the whole encoder can be expressed as:

$$(\mu, \log \sigma^2) = \Phi(x_t + 1, x_t)$$

During training, the reparameterization trick [KW22] is utilized by scaling a uniformly distributed sample to the distribution and, thus, circumventing the non-differentiability of the stochastic model. With the random noise sample $\epsilon \sim \mathcal{N}(0, 1)$, we can rescale it to generate a sample of the latent distribution by computing $z_t = \epsilon * \sigma + \mu$. Thus, the possibility for backpropagation of information to the encoder remains possible. The resulting latent action control vector z_t has a length of 32.

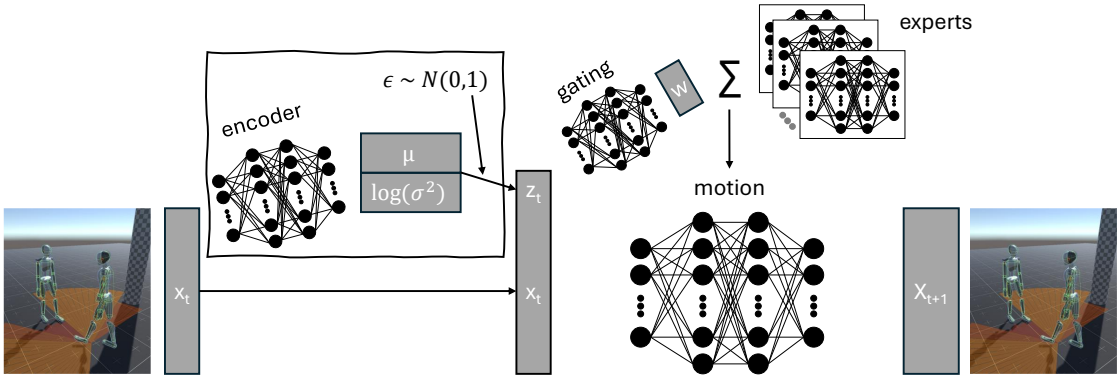


Figure 4.26: Detailed structure of the motion VAE network. The encoder is only utilized in training. During inference, the animation generation is controlled via the latent variable z_t. The decoder is a MoE model utilizing the concatenation of the latent vector and the frame x_t as an input for the gating and motion network to predict the next frame x_{t+1}.

The decoder is a MoE model [Zha+18] using six expert networks and a single gating network. The input to the decoder is the concatenation of the latent z_t and the previous pose x_t. The gating network Ψ is a three-layer fully connected network with ELU activations and a similar computation as the encoder shown above. The hidden layer size is 64, and the softmax operator is utilized to ensure that the sum of the expert weights $e \in \mathbb{R}^6$ equals to one:

$$e = \text{softmax}(\Psi(x_t, z_t))$$

For each iteration, the expert weights are used to create a specialized network Ω with weights W_i^ω and biases B_i^ω using a weighted sum of the expert networks defined by their individual weights $w_{j,i}$ and biases $b_{j,i}$:

$$W_i^\omega = \sum_j^6 e_j \cdot w_{j,i}^\omega \quad \text{and} \quad B_i^\omega = \sum_j^6 e_j \cdot b_{j,i}^\omega$$

The resulting weights are utilized to execute the three-layer decoder with layers similar to the encoder and gating network using the ELU operator and a latent size of 512

$$x_{t+1} = \Omega(x_t, z_t)$$

The network weights are optimized with stochastic gradient descent using the Adam algorithm [KB17] in mini-batches of 64 data points. Learning rate decay is applied starting at 1e-4 and decaying to 1e-7. The training data is normalized using z-score, and group normalization is applied [WH20] with the groups being the character displacement, joint positions, velocities, and rotations. Scheduled sampling [Ben+15] is utilized to train for autoregressive motion prediction in three stages: (i) 20 epochs with ground truth data only (teacher stage), (ii) 20 epochs with increasing autoregressive prediction (ramping stage), (iii) 100 epochs in pure autoregressive mode (autoregressive stage). In each stage, the network performs eight regression steps.

The reconstruction error is penalized with the mean squared error (MSE), and the variational autoencoder is constrained via β-VAE training using Kullback-Leibler (KL) loss ($\beta = 0.01$).

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{t=1}^{N}\left(x_{i+1} - \Omega(x_t, \Theta(x_{t+1}, x_t)\right)^2$$
$$\mathcal{L}_{KL} = -0.5 \cdot \sum_{t=1}^{N}\left(1 + \log\sigma_t^2 - \mu_t^2 - \sigma_t^2\right)$$
$$\mathcal{L} = \mathcal{L}_{MSE} + \beta \cdot \mathcal{L}_{KL}$$

The motion model was trained on an Nvidia A100 GPU with 80 GB VRAM, and training was concluded after approximately six hours. The motion model was exported in the open neural network exchange (ONNX) format and integrated within a Unity 3D environment using the Microsoft ONNX Runtime library.

Action Policy

For the training of the action policy, the Unity ML-Agents Toolkit [Jul+20] is utilized. The sensors and motion generation pipeline are implemented within a custom Unity environment. Thus, the policy training does not change the motion model's network weights. The default navigation mesh implementation inside of Unity, which only considers static but not dynamic objects (like the opponents), is utilized to provide the navigation input, consisting of the directional velocity $\vec{v_t} = (v_t^x, v_t^z) \in \mathbb{R}^2$ and direction from the agent to the closest point on the ideal path truncated to a uniform length in root local space $D_t = (d_t^x, d_t^z) \in \mathbb{R}^2$. With the input of the equipped perception system $(S_t|C_t|V_t)$ and an observation flag, indicating whether an obstacle is perceived $O_t^{S|C|V} \in \mathbb{R}$, the perception vector is

$$s_t = \left\{x_t, v_t^x, v_t^z, d_t^x, d_t^z, (S_t|C_t|V_t), O_t^{(S|C|V)}\right\},$$

Several rewards and penalties are applied to control the training of the action policy and form the cumulative reward R_t for every frame. The individual components of this reward are described in more detail subsequently. All sub-rewards are bound and scaled to proportion. Hence, the cumulative reward per frame is primarily in the $[-1, 1]$ range.

$$R_t = R_t^{ppr} + R_t^{dir} + R_t^{tar} - R_t^{div} - R_t^{prox} - R_t^{st}$$

Path Following Rewards. The following rewards are applied to control the agent to follow a predefined trajectory on individual subgoals.

- **The path point reach reward** R_t^{ppr} is a one-time reward of 10 when the agent gets closer than 0.5 m to the next path destination point.

- **The direction reward** is applied to encourage walking directly to the target by reinforcing the factual velocity $\hat{v_t}$ and factual movement angle $\hat{v}_t^a$ to be close to the desired velocity v_t and movement angle a_t, bound by $[f_l^{dir}, f_u^{dir}]$ with the empirically derived values of $[-0.15, 0.25]$.

$$R_t^{dir} = \left(f_u^{dir} - f_l^{dir}\right)\left(\left(\frac{\cos\left(\hat{v}_t^a - v_t^a\right)}{2}\right)^4 \cdot e^{-|\|\hat{v}_t\| - \|v_t\||}\right) + f_l^{dir}$$

- **The target approach reward** is a continuous reward to encourage moving towards the next goal by computing the difference $l_t^g = \|g_{t-1} - r_{t-1}\| - \|g_t - r_t\|$ between the previous and current distances to the destination point g_t. To bound and scale the reward to the range of $[-0.25, 0.25]$, a rational asymptote A with the sensitivity factor $\alpha^{tar} = 400$ is applied:

$$A(x, \alpha) = 1 - \frac{1}{1 + \alpha * x}$$
$$R_t^{tar} = 0.25 \cdot \frac{|l_t^g|}{l_t^g} \cdot A\left(|l_t^g|, \alpha^{tar}\right)$$

- **Path divergence** is penalized if the agent diverts from the dedicated path. It is again bound with a rational asymptote and scaled to a range of $[0, 0.2]$:

$$R_t^{div} = 0.2 \cdot A\left(\|d_t\|, 1\right)$$

Agent Avoiding Rewards. During avoidance maneuvers, the agent should not collide with the opponent but keep as close to the original trajectory as possible. Simple one-time collision penalties and full-body capsule colliders were insufficient and did not create the desired results. Hence, multiple colliders were created for individual body parts (e.g., legs, arms, torso, etc.).

- **The proximity penalty** is computed using the minimal distance between two body parts of opposing agent s_{ij}. This distance is computed for every collider in the agent body in the set $i \in C^{body}$ to every scene object within the radius r around the agent in the set $j \in S^{obj}$ containing the body colliders of other agents. While this is computationally more expensive, it is only required during training and not during inference. Quadradic decay is applied to the loss to avoid over-penalization of far away bodies:

$$R_t^{prox} = 1.5 \cdot \frac{1}{1 + 10 \cdot \left(\min s_{ij}\right)^2}$$

- **The shoulder turn penalty** is used to encourage upper body movement for avoidance by projecting the agent's shoulder positions on a line $\vec{l_t^s}$ orthogonal to the movement direction on the ground plane. The distance is first scaled proportionately to the maximum $L_{\max}$ and then linearly to the range [0,0.5].

$$R_t^{st} = 0.5 \cdot \min\left(\max\left(1 - \frac{|\vec{l_t^s}|}{L^{\max}}, 0\right), 1\right)$$

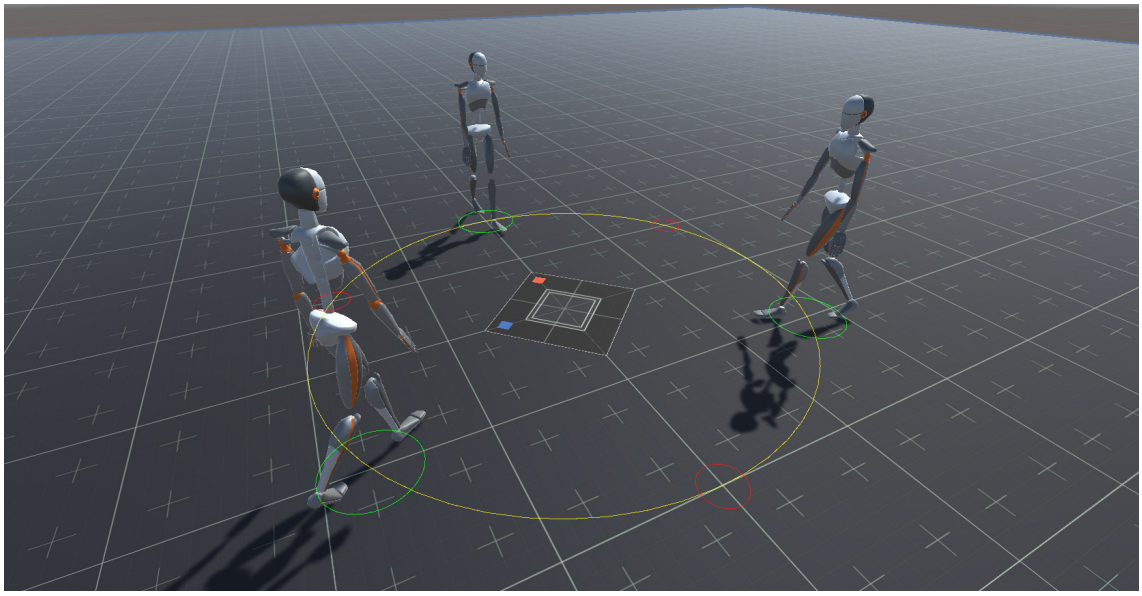


Figure 4.27: Concurrent avoidance environment with three agents. The agents are spawned on the green start circles and are trained to reach the red target circles on the opposite side while avoiding collisions in the center. Reprinted with permission from [Akm+25] © 2025 IEEE.

Training Environments

Three different training environments are utilized. The first **walking** environment contains a single agent. A random destination point is selected, and upon reaching the point, the agent does not receive any command for three seconds to encourage idling. After the idle period or the maximum number of steps is exceeded, the agent is respawned at a different location, and a new target point is sampled. In the **concurrent avoidance** environment, multiple agents are spawned on a circle with the target point on the opposite side, forcing the agents to pass the circle's center and thus create collision events. The number of agents is varied between $[2, 4]$, the radius of the circle between $[2\text{ m}, 5\text{ m}]$, and the heading angle between $[0°, 180°]$ with respect to the center of the circle. Figure 4.27 shows an example of this environment. Although many collisions occur in this scenario, it is fairly predictable. A **random avoidance** environment with five agents walking towards random goal positions in a confined space is utilized to reduce predictability while not containing as many collision events as the concurrent avoidance movements. Experience is gathered from 20 agents in parallel in multiple environments to speed up training. The most optimal combination of environments was four walking environments (same configuration), four concurrent avoidance environments (two, two, three, and four agents), and one random avoidance environment with five agents.

Table 4.5: Training time, number of training iterations, and utilized GPU for the different models. (Table originally from [Akm24])

Model	Time (hours)	Iterations	GPU (NVidia)
Blank (no perception)	10:40	20e6	V100 40 GB
Sector	12:36	20e6	V100 40 GB
Voxel	14:50	20e6	V100 40 GB
Sector & Voxel	14:50	20e6	V100 40 GB
Camera	27:32	10e6	A100 80 GB

4.5.4 Evaluation

Policy Training Time

A single policy was trained for each perception type for 20e6 iterations, except for the camera perception model, which exceeded more than double the training time of the other models after 10e6 iterations. Training took between 10:40 to 27:32 hours (see Table 4.5 for details).

Runtime Performance

The average frames per second (fps) during inference were recorded in the deployed application with no fps cap and disabled vertical synchronization. The evaluations were executed on a mid-range gaming laptop (Intel i7-6700HQ 2.6GHz, NVIDIA GTX 1070 Laptop, 16 GB RAM). Even with a very small input size of 32x32 px, the depth camera perception agent does not scale. With more than nine agents in one environment, the performance drops below the required 60 fps for interactive applications. All other perception models do scale, and especially with the sector perception agent, it is possible to execute more than 20 agents in parallel at over 60 fps. Even on a high-end gaming PC (AMD Ryzen 9 16 Core CPU, NVIDIA RTX 3080 GPU, 64 GB RAM), the camera agents' performance drops similarly, while 25 sector perception agents run at 135 fps. As this makes the camera-based agent unusable for the target application, it has not been further evaluated. For more details, see Figure 4.28.

Visual Evaluation

All agents show high-quality locomotion animations and can generally steer to a goal position. As expected, a blank agent without the ability to perceive opponents collides with other agents frequently, sometimes showing random evasion movements, resulting in an initially missed target and circling around the target. The sector perception agent displays avoidance movements and passive upper body rotations to pass by an opponent. It shows that learning to avoid others through spatial body movements is feasible rather than relying on pathfinding skills only. However, there are still instances when a collision occurs, and the system cannot adequately react to suddenly blocked passages. The voxel perception agent slows down and tries to avoid opponents upon perceiving them, but not always successfully. In the case of four opponents, the agent tends to slow

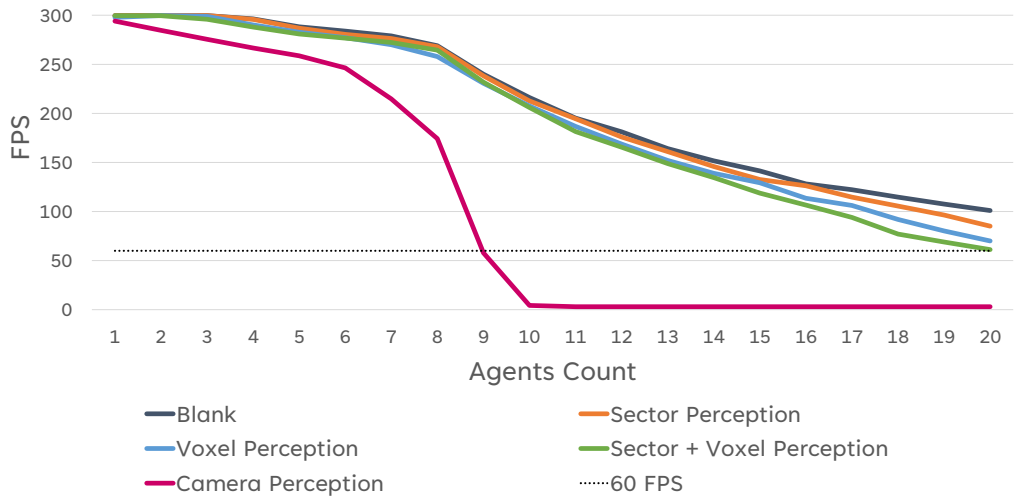


Figure 4.28: Runtime performance for different numbers of agents and perception models running in parallel on a mid-range gaming laptop. Reprinted with permission from [Akm+25] © 2025 IEEE.

down significantly, most likely due to more voxels being triggered in front of the agent, signaling it to stop and wait for a clear path. The combination of sector and voxel perception is unstable and not very agile. While the assumption was that the combination of both perceptions would benefit the agent's behavior, it does combine its shortcomings as well. In some cases, the body is paralyzed and stops moving, especially during encounters with an opponent. In addition, this model showed the worst behavior regarding reaching the goal position.

Walking Performance

An objective evaluation of the walking performance was conducted by positioning the goal between two to five meters of the agent and rotating the agent 0, 90, or 180 degrees to the goal direction. In total, 1200 trials were executed for each model, and the conditions were balanced over the trials. While the blank and sector agents reached the goal in all trials within the given time window, the voxel agent missed three targets, and the combined sector and voxel agent missed seven. However, not all models are equally efficient in reaching the target, and sometimes, they miss the goal on the first try. Figure 4.29 shows each agent's average number of frames to reach the target based on the starting distance. While the performance is comparable on average for the sector and blank agents, there is some variance, especially on short distances, primarily due to initially missed targets and subsequent retries to reaching the target by 'circling' around the target position multiple times due to a limited turn angle.

Avoidance Performance

For the objective evaluation of the avoidance performance, the concurrent avoidance environment is utilized with a circle diameter of four to ten meters and two to four agents. With 100 trials per configuration and each agent within the scene recording data, a total of 3600 trials were executed.

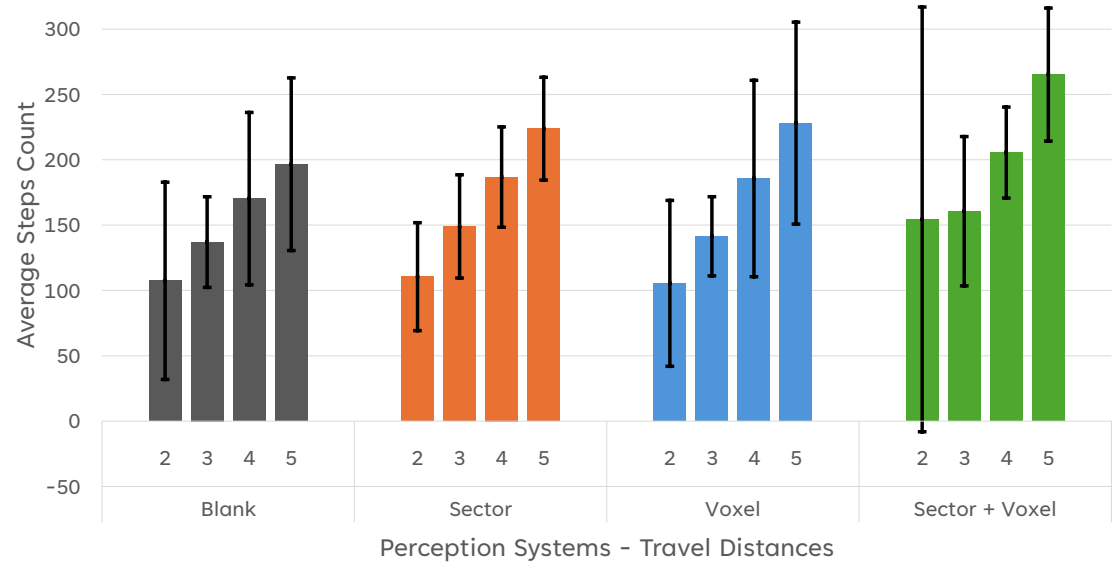


Figure 4.29: The average number of steps required to reach the goal. Error bars denote one standard deviation. Results are grouped for the agent perception system and the initial distance to the goal. Reprinted with permission from [Akm+25] © 2025 IEEE.

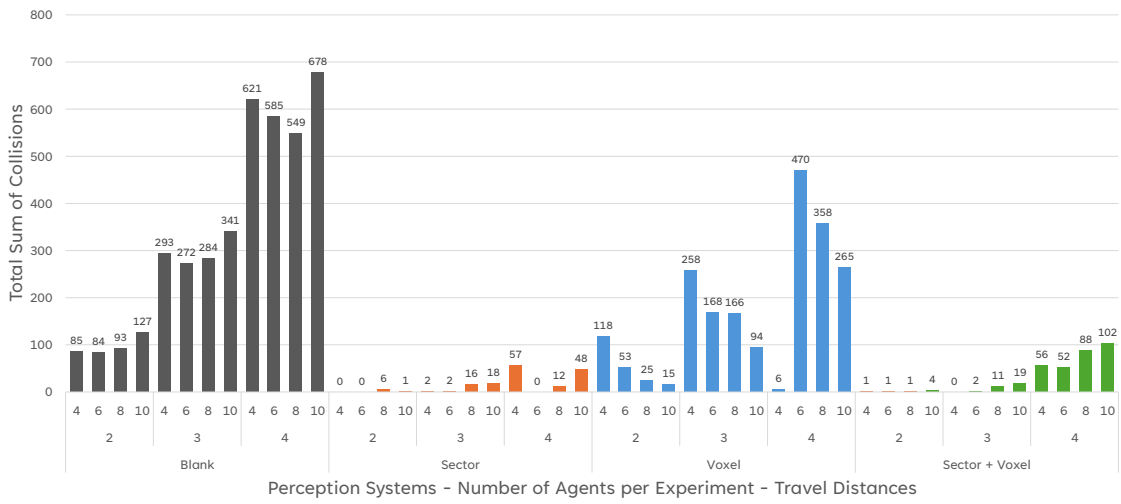


Figure 4.30: Total sum of collisions per perception agent. Agents using the sector perception produce the fewest collisions. Reprinted with permission from [Akm+25] © 2025 IEEE.

While the sector perception agent is able to reach the target in all but one of the trials, the blank agent misses reaching the target within the time limit in 22 trials, the combined sector and voxel agent in 13 trials, and the voxel perception agent in 74 trials.

Even without perception, two agents do not necessarily collide. Small deviations in trajectory, acceleration, and velocity can result in two agents missing each other. On the other hand, agents might collide more than once, especially if the target is initially missed and the agent circles to try it a second time. In the case of a model reaching the target perfectly every time and all agents bumping into each other exactly once, a total of 4000 collisions would be recorded. Collisions are recorded with body colliders, matching individual body parts, not by a simple capsule collider around the character. The blank agent without perception recorded 4012 collisions during the experiment, including some iterations without collisions and some with multiple collisions. The sector perception agent performed best, with a total of 162 collisions. The voxel perception agent performed poorly with 1996 collisions, and the combination of both sensors did not benefit either (337 collisions). Figure 4.30 shows the detailed distribution of all collisions for each agent type, the number of agents, and the starting distance. There are no consistent differences between the starting distances, but more agents in the scene generally result in more collisions. Further in-depth analysis shows that the minimal interpersonal distance in the sector perception agents is 0.427 m, being considerably lower than a typical capsule collider and thus allowing closer passing maneuvers. Thus, our target model using the sector performance reduces the number of collisions by over 95 % while being able to reach a target position consistently.

Hyperparameter Evaluation

During the development, several hyperparameters were fine-tuned and tested.

1. While 10e6 training iterations yield sufficient performance in walking, it is insufficient for avoidance movements, and 20e6 iterations improve results significantly.
2. Number of hidden layers of the motion decoder: avoidance skills are lacking with two layers, and four layers require significantly more training without a better performance.
3. During reinforcement learning, 256 exploration steps provide faster convergence to the desired behavior than larger step counts (e.g., 1024) without reducing performance.
4. The path point reach reward R_t^{ppr} has only minor impact and could most likely be removed
5. If the path divergence penalty R_t^{div} is weighted too high (>1), the model sticks to the path too closely, resulting in more collisions.
6. If the proximity penalty R_t^{prox} is weighted too high (>2), agents choose very convoluted trajectories and divert significantly from the planned trajectory to avoid collisions.

4.5.5 Discussion

This section presents a new approach for generating character avoidance movements in multi-agent environments using an autoregressive conditional variational autoencoder controlled by an action policy trained with reinforcement learning. The approach requires only a few minutes of motion capture data of a single actor to effectively avoid other characters with minimal implementation effort and almost no adjustments for the navigation planner. Different perception modes to perceive a variable number of opponents have been evaluated with respect to their efficiency and effectiveness. Using a novel perception mechanism, the target model can efficiently simulate over 20 agents, showing avoidance characteristics in our experiments.

Limitations

The action policy is not yet perfect. It does not always reach the goal point accurately, resulting in undesired behavior (e.g., circling around the goal position). While it displays only a few collisions in our experiments, the model struggles to maintain this performance in highly crowded environments.

Future Work

While the proposed approach was able to synthesize realistic avoidance movements like upper body rotations, it did not always utilize them. Adding more annotations to the data (e.g., frame-wise annotation with an 'avoid' flag) could establish a more explicit control over the motion, and different avoidance styles could be introduced. Combining a classical motion planner as a hybrid system would be highly encouraged in a target application to make the system more robust and always ensure avoidance, even if it falls back to classical capsule colliders. By fine-tuning the parameters of the reward function, different types of agents (more aggressive, more submissive, right-handed, left-handed avoidance, etc.) could be trained using the same or a different motion source to increase variability inside a crowd and match the parameters of the agent behavior. Lastly, the sensors only contain snapshots of the current state. Incorporating more temporal information, like the directional velocity of individual sectors, voxels, or pixels, could improve the utility of the environmental information. Alternatively, compressing temporal sequences to a latent dimension with an encoder and passing the latent code to the network could be beneficial. This would provide more information regarding the importance of the sensor information for the ego-character (e.g., a dynamic object moving towards or away from it).

4.6 Chapter Discussion and Future Work

4.6.1 Discussion

It is important to include natural variations in motion to create realistic and plausible animations of multiple agents in a scene. Even if pedestrians have a different outer appearance, if their motion is monotone and repeated by everyone in the crowd, it is neither realistic nor useful. Natural variations are required to avoid overfitting the autonomous vehicle on individual motions and learn the correct cues indicating a crossing intention. Two approaches for generating this variation were created. First, a system was created to replicate the natural style between female and male pedestrians, allowing an individual configuration of agents and increasing the inter-individual variation. The approach was built upon a MoE architecture and utilized explicit control of the locomotion (walking direction and velocity). The second approach was built upon a similar architecture. It included variation within the network layers by adopting concepts of VAE and increasing the intra-individual variation of a single individual. In the last section of this chapter, a novel approach for pedestrian avoidance animation is presented, built upon a generative cVAE controlled by an action policy trained with RL. While the focus of this study was not on variation, the utilized technology is generating variational motion by design.

Next, the virtual agents need to be able to control the animation with sufficient accuracy. As an agent simulation framework controls the pedestrians, there are two control modes. Direct control provides direct input in every frame, for example, in which direction to walk or where to look. While human players are excellent at compensating for any mistake in the animation system and learn the system's dynamics unconsciously, a directly controlling agent usually has its intrinsic model for the dynamics (e.g., acceleration profile, turn rate, etc.). For example, an agent could model the trajectory behavior, including the acceleration behavior and the natural hip sway during idling and walking, simulating the body's inertia. The agent could, however, be simulated as well as a simple cylinder occupying a general space and moving instantaneously. In both cases, the pedestrian animation system is highly unlikely to display exactly the same dynamics. On the other hand, agent models considering an indirect control provide only sparse control information. For example, trajectory goals or actions, that should be performed at a specific spatio-temporal location. In these cases, the animation system is comparatively free in its own choice of how the actions are precisely executed.

The models in Section 4.2 and Section 4.3 assume a human controller or an agent with direct control. However, they only enable the control of the locomotion direction, velocity, and style. In Section 4.4, different studies were conducted to extend the capability of the models and enable the control of more actions. In particular, the motion representation was extended to utilize future trajectories of body parts to control their movement with higher precision. While the control of gait-correlated actions, like the gaze directions, can be generated with sufficient quality and only minor implementation effort, it was hard to control other limbs due to the correlation of poses

within the datasets. Providing additional control of the feet improved stepping on stairs but did not enable accurate control over the feet. In the case of the upper body, providing additional control over the hands did improve the accuracy, but it is still not at the level required in many applications. Therefore, we switched to a new architecture, utilizing a cVAE combined with an action policy trained with RL. With this model, the controller can be configured by defining the loss functions utilized to train the action policy. Besides standard locomotion tasks (walking to a goal point, walking in a specific direction, etc.), the upper body was rotated to avoid other pedestrians without the intensive implementation and data processing effort required with the past methods.

4.6.2 Future Work

Generative models are the future of motion simulation. cVAEs are fast with variable motion output and, combined with reinforcement policies for action control, are a lightweight, fast, and adaptable solution for many animation classes. Control policies can be trained with a direct control (directional velocity input at every frame) or indirect control (target position to be reached). When considering the large amount of data captured in the studies outlined in Chapter 3, cVAEs offer additional capabilities when exploiting the condition further. In the approaches utilized in Section 4.5, the past pose defines the condition. In future studies, more information should be added to the condition to increase the variability and applicability of motion models. Following the studies of Du et al. [Du+19], adding a style variable to the condition enables the model to generate animations in that specific style. Thus, adding meta-information about our study participants (e.g., gender, age, height, nationality, etc.) or behavioral parameters (e.g., cautiousness, politeness, engagement, etc.) should enable the system to learn the natural differences between motions depending on these variables and generate matching animations.

Diffusion models are a new class of generative AI that can be applied to motion simulation. They are used to denoise a whole trajectory of motion, potentially enabling more controllable and accurate animations. They are not necessarily causal methods, and information on potential future states can leak to the generation of the next frame. Take, for example, the task of reaching an accurate space-time location to trigger a collision at the first quarter of the front of the approaching vehicle. If the model is controlled in a recurrent, online fashion with a reinforcement policy, it can only guide the animation towards the goal in every frame. Thus, it can neither control the exact time when the target is reached nor ensure it is reached at all. This prevents standard post-correction with inverse kinematics and animation blending and can result in errors. In our work presented in Section 4.5, agents missing the goal position tend to circle the goal for 0.5 - 1.5 rounds until they are close enough to trigger the corresponding event. In future work, new approaches should be investigated, which utilize diffusion networks to optimize the animation to ensure that spatio-temporal constraints are always matched.

Chapter 5
Visualizing Virtual Pedestrians

Research on pedestrian agent and motion simulation is usually conducted in custom environments like Python scripts [Lin+20], OpenGL (Open Graphics Library) environments [HKS17]), or utilizing research frameworks like the skinned multi-person linear model (SMPL) [Lop+15]. Other approaches are well integrated into a specific game engines (e.g., Unity [Zha+18; Sta+21]). Particularly for motion simulation, models are usually observed in isolation, assuming the model generates all animations on a pre-known skeleton. While these aspects facilitate research in animation technology as they reduce unnecessary implementation and integration overhead, they complicate the deployment of new methods in other applications like ADAS driving simulators, that are not necessarily compatible with the motion simulation environment.

During this dissertation, the natural disasters caused by climate change, a worldwide pandemic, and the unprovoked attack and the subsequent occupation in Ukraine have resulted in significant changes to the market and the hesitancy of the automotive industry to invest in new technology. Hence, existing approaches and methods are preferred, and new technology should be integrated into existing pipelines with minimal effort. Existing industrial pipelines utilize a heterogenous set of software solutions from software based on computer-aided design (CAD) like Car Maker[29] to novel rendering engines like NVIDIA Omniverse[30] or game-engine-based simulators like CARLA [Dos+17]. Native integration into this heterogeneous environment and utilizing existing material, like character models, is complicated and effortful. To overcome this challenge, novel methods for exchanging motion information between simulation software and for transferring motion to new virtual characters are required.

This chapter focuses on how pedestrian agent movements can be integrated into ADAS simulation environments and different virtual pedestrian characters. Section 5.2 focuses on retargeting motion to different humanoid characters in order to animate a diverse crowd with a single animation model. In Section 5.3, the MOSIM Framework is presented, a modular framework for the combination of different motion simulation approaches in different simulation engines, like autonomous driving simulators. But first, the related work is outlined.

[29] https://ipg-automotive.com/de/produkte-loesungen/software/carmaker/
[30] https://www.nvidia.com/de-de/omniverse/

5.1 Related Work on the Visualization of Motion

5.1.1 Digital 3D Engines

Two types of 3D software can be differentiated: Computer-aided design (CAD) software, in which objects are described with mathematical functions, and surface-based renderers, in which sets of vertices and faces define surfaces to be rendered. CAD software is highly optimized for defining 3D objects in industrial engineering. This ranges from individual parts, which should be constructed, to individual sub-assemblies inside a car or the complete car in itself. With CAD software, these components can be accurately described, automatically integrated, and manufactured in production. However, these systems usually do not natively support humanoid simulation. Highly specialized humanoid simulation models, such as the IMMA manikin[31] or the RAMSIS model[32], can be utilized to some degree but are neither freely available nor have a large community of developers to support new features and content.

3D computer graphic engines are utilized in creative environments, movie production, and, most of all, game development. Unlike CAD software, the primary focus is on the visual quality of the rendered images rather than geometric accuracy. Graphic engines directly support the animation of humanoid characters. In addition to specialized but closed-source applications, the utilization of accessible game engines like Unity and Unreal Engine is rising. These game engines contain tools for the animation of movements of pedestrian agents and their behavior using scripts and blending trees. Although most publicly available animations and behavior models are specifically tailored for the gaming domain, the tools are well documented and can be applied to create suitable motion models for industrial applications. Game engines can display high-fidelity textured meshes but do not natively support CAD data or the mathematical description of complex surfaces and volumes. Manual integration of CAD data is possible but tedious, and additional software for hardware-in-the-loop testing is required, such as realvirtual.io[33], custom ROS [Qui+09] applications, or using OPC-UA endpoints [EM11] (endpoints for the Open Platform Communications - Unified Architecture). New plugins, like Pixyz[34] in Unity, and Datasmith[35] in Unreal Engine assist in integrating static CAD data to game engines. Recently, NVIDIA presented the Omniverse platform[36], combining a powerful real-time raytracing engine with computational modeling and game engines. As the tool is very new, its integration into existing workflows remains to be seen.

In the scope of this dissertation, demonstrators, experiments, and tools were developed with the Unity engine, Unreal Engine 4, and Unreal Engine 5. The Blender editor was utilized for 3D model design.

[31] https://industrialpathsolutions.comn/ips-imma
[32] https://www.human-solutions.com/en/products/ramsis-general/index.html
[33] https://realvirtual.io/
[34] https://unity.com/products/pixyz
[35] https://www.unrealengine.com/en-US/datasmith
[36] https://www.nvidia.com/de-de/omniverse

5.1.2 Human Motion Simulation Frameworks

The simulation of humanoid motions is an extensively researched topic, as outlined in Chapter 4. While some methods focus on individual body parts (e.g., arm movement, finger movement, etc.), others consider full body motions (e.g., locomotion, sports). They all have in common that they consider their own representation of motion. While some methods are partially integrated into existing game engines, the combination of different methods is insufficiently addressed in research.

To create animation sequences on a virtual character, there are different tools like MotionBuilder[37], Maya[38], Poser[39], DAZ 3D[40], and Blender, in which characters can be posed, and keyframes can be generated (see Section 2.5 for more details). Motion capture data (see Section 2.7) is often utilized to generate a natural prior but needs to be post-processed and adjusted before being used in an actual application. All game engines (e.g., Unity, Unreal Engine, Godot[41], Panda3D[42], etc.) have their own animation system, in which animation data can be loaded, usually via the FBX file format (Autodesk Filmbox; *.fbx)[43], and blend trees can be generated to blend animation sequences based on control input. Unity's closed-source animation model, Mecanim[44], is based on a simplified muscle structure for full-body animations. Unreal Engine has recently published Metahuman[45], which simplifies facial capturing with smartphones and animation on virtual characters. Unreal Engine has additional native support for behavior trees to model and simulate the behavior of virtual agents. Game engine animation systems are focused on very efficient animation of characters relying primarily on predefined animation sequences. The combination with other animation tools (e.g., neural networks) is not natively supported. Although all game engines utilize a finite state machine (blend tree) for blending animations, their internal representations are incompatible.

In contrast, there have been academic approaches for unified simulation frameworks in the past. The most prominent of these is the Behavior Markup Language (BML) [Kop+06], an XML-based (extensible markup language) formal language to describe the enactment of social actions on a detailed level. Specific body parts can be targeted (e.g., head), and individual movements (e.g., giving a handshake) can utilize synchronization points to orchestrate more complex behavior (e.g., nodding while giving a handshake). Furthermore, conditions and events can be utilized to concatenate motions, and duration parameters can limit the duration of continuous actions. An example can be found in Listing 5.1. BML is used in different software frameworks.

The SmartBody [Thi+08] framework for the simulation of embodied conversational agents allows to specify actions that are then animated with animation controllers via the BML language. Its main focus is on the synchronization between speech and motion to simulate conversational

[37] https://www.autodesk.com/products/motionbuilder
[38] https://www.autodesk.com/de/products/maya
[39] https://www.posersoftware.com/
[40] https://www.daz3d.com/
[41] https://godotengine.org/
[42] https://www.panda3d.org/
[43] https://www.autodesk.com/products/fbx/overview
[44] https://docs.unity3d.com/Manual/AnimationOverview.html
[45] https://www.unrealengine.com/en-US/metahuman

Listing 5.1: Example BML code to wave to another person.

```
<bml>
    <body id="b1" posture="stand" />
    <gesture id="g1" type="wave" target="human2" duration="4.0" start="b1:end" />
</bml>
```

agents, and it is integrated into different game engines (e.g., Unity, Ogre). Heloir et al. [HK09; HK10] propose an animation layer in their embodied agents behavior realizer (EMBR), that enables a closer control of the animations and that can be used as an intermediate layer between BML and the actual motion. Users can define more specific parameters like goal positions, blending parameters, and individual joints affected by a motion in EMBREScript, a Pascal-like scripting language. By default, the system allows forward and inverse kinematics. Other parameters influence the temporal and spatial extent, the power, and the fluidity of a base animation. Shoulson et al. [Sho+14] introduce an agent development and prototyping testbed (ADAPT), separating behavior modeling, navigation planning, and animation into three components. The character animation is generated by *choreographers*, which perform the computation on an invisible clone of the target skeleton. The output of the choreographers is combined by partial blending of poses (e.g., only upper body, only right or left arm, etc.) in a *coordinator* component. The authors provide examples for an animation-based locomotion choreographer, an IK-based gaze choreographer, a procedural optimization-based approach for reaching, a ragdoll model for physical interactions, and a SmartBody [Thi+08] integration. ADAPT was realized within the Unity game engine.

The animation systems outlined above focus on embodied agent simulation and simplified animation control for domain experts with limited programming knowledge. Other systems (e.g., [Ree+06]) focus on the manufacturing use-case, including ergonomic analysis. Regarding the quality of animations, these systems are easily outperformed by the game engine integrated animation systems. While some of the systems like SmartBody [Thi+08] and ADAPT [Sho+14] are modular and allow different algorithms to generate the animation, they are not modular in the sense that animations can be generated in a completely different process or even computed on a different device. In addition, the intellectual property of the animation module cannot be preserved.

Herrmann et al. [Her+19b] propose a management system to store, annotate, and prepare motion capture data for model training. The system includes model training and distribution. The deployment server provides functionality to specify constraints via a JSON data structure and a network protocol like TCP. The server utilizes a motion graph implementation to compute the corresponding animation and return it to the remote client. Herrmann [Eri25] demonstrates a further extension to enable different latent animation models, like neural network-driven animations. However, the approach remains a single application implemented in Python, missing the ability to include and utilize additional animation systems and game engines as an animation source. Nevertheless, his work has been an essential conceptual basis for developing the MOSIM Framework.

5.1.3 Online Motion Transfer and Retargeting

The animation of a character is tied to its underlying animation system, usually a hierarchical kinematic skeleton. By retargeting (or transferring) the motion between two characters, animations can be reused to save time and effort. In existing game engines, the retargeting of animation sequences is already well established. While Unity utilizes an internal, muscle-based animation system (Mecanim) and requires direct retargeting to its internal structure when importing an animation, Unreal Engine is more liberal and supports arbitrary kinematic skeletons. However, in Unreal Engine a separate retargeting function must be configured for any two different skeletons, resulting in a many-to-many relationship.

Gleicher et al. [Gle98] propose one of the first approaches for motion retargeting by annotating animations with space-time constraints and applying inverse kinematics (IK) to transfer animations to differently sized characters with the same hierarchy. Monzani et al. [Mon+00] extend this line of thought and propose a method for retargeting animation sequences between characters of different hierarchies and scales by mapping the animation to an intermediate skeleton representation first and applying IK after retargeting to reach annotated end-effector constraints. While these basic concepts are now implemented in production environments, they cannot be directly applied to real-time retargeting.

For online retargeting, Feng et al. [Fen+14] propose an automatic process of joint mapping, skeleton alignment, motion data transfer, and constraint enforcement. The similarity between humanoid skeletons is leveraged to employ heuristic methods in identifying a joint mapping. Joints are then rotated to match the global direction in space to align both skeletons. By storing the rotational difference, motion can be transferred during runtime. Additional IK is applied to enforce constraints computed on the preshared animation before retargeting.

Particularly in games, an indefinite amount of morphologies for humanoids and creatures is possible, such as in the game "Spore"[46], where a player can create new lifeforms of arbitrary shape and size. To retarget motion between these morphologies in real time, Hecker et al. [Hec+08] propose a new system utilizing specialized and annotated key-frame animations and static preprocessing on the target avatar (e.g., checking for an upright spine, counting the number of legs, etc.). A particle IK solver generates the pose during runtime, and a secondary motion ("jiggles") is applied to all non-IK joints. While the approach shows great results for arbitrary morphologies, it requires intensive pre-processing of the animations to reach specific spatial targets, and the approach was not demonstrated for human characters.

Even though humanoid models have very similar morphologies, the hierarchies and number of joints can differ. New animation data must be created if an animation source contains significantly fewer joints than the target character. While IK solvers can compensate a mismatch of a few joints sometimes, more elaborated approaches need to be employed if the number of joints differs more greatly. Reda et al. [Red+23] propose an RL-based approach operating on a physical body model

[46] https://www.spore.com/

to retarget from VR-sensor data containing only head and hand information to different virtual characters. Their training pipeline, however, relies heavily on a large-scale motion capture database and a kinematic retargeting of the training data to the known morphologies before training time. It thus cannot be applied to novel morphologies during run-time.

There are different neural-network-based retargeting approaches. Villegas et al. [Vil+18] propose a neural architecture for retargeting between differently scaled characters with the same skeletal hierarchy. Their approach consists of a recurrent neural network (RNN) with a differentiable forward kinematics (FK) layer. While an adversarial cycle consistency loss increases performance and temporal stability, the overall approach for retargeting barely outperforms a direct copy of quaternions. The approach was extended by Villegas et al. [Vil+21] to incorporate the geometry of the character and prevent the skin's inter-penetration after retargeting. While the approach did not only look more plausible in a user study, it significantly reduced inter-penetration and joint position errors. Aberman et al. [Abe+20] utilize the homomorphic nature of humanoid skeletons to train an encoder-decoder architecture that encodes a pose from a source skeleton to a simplified skeleton structure and decodes it to the target skeleton structure. However, their approach requires an individual animation dataset for each skeleton to retarget. Hu et al. [Hu+24] also utilize the homomorphism by training latent spaces for each body part (e.g., left arm, right leg, etc.), effectively retargeting individual body parts rather than a complete pose. While the results of neural-network-based approaches appear natural, they do not ensure constraints and further post-processing (e.g., IK) is required to ensure that end-effectors are positioned correctly.

Besides the differences in skeletal hierarchy, characters can differ in size and shape. Accurately retargeting motion between different types of characters (e.g., adults to children, thin to fat, small to tall) requires more than just the motion transfer and is more similar to style transfer rather than animation transfer. Dong et al. [Don+17] apply dynamic scaling laws for adult-to-child retargeting and evaluate the result with a perceptual study. Based on these insights, Dong et al. [Don+20] propose a Cycle-GAN-based approach to retarget animations from adults to children by training the GAN on a small set of unpaired motion data for adults and children. There are some approaches to creating highly detailed skin and muscle deformations [Suj+18], directly operating on the mesh without a skeleton [Lia+22; Wan+23], or introducing custom rigging solutions [Bha+12]. Directly operating on a mesh level requires specific mesh types or character models and, thus, is difficult to use for integration into ADAS simulations.

5.2 Configurable, Bi-Directional, and Real-Time Retargeting

In pedestrian simulation and other animation domains like manufacturing simulations and games, it is essential to simulate diverse characters. Retargeting describes the motion transfer from one human representation to another, making it possible to use one animation source to drive many different-looking characters. In this work, we focus on retargeting skeleton-based animations between characters with different skeleton sizes and skeleton hierarchies.

While some approaches consider retargeting as the harmonization of different motion capture datasets to the same distribution [Mah+19], others consider the transfer of a single motion sequence from one representation to another [Mon+00]. The first approach helps increase the amount of training data, and the second is common practice in classical animations. However, both views on retargeting are offline in nature, assume full access to the whole sequence before retargeting, and require manual post-processing or large training datasets. Thus, they are not applicable to real-time retargeting of apriori unknown motions and representations, which is particularly important for neural network-driven animations. On the other hand, online or on-the-fly retargeting [Fen+14] does not rely on pre-exchanged animation data. Still, the retargeting function must be adjustable by a human, as different skeletal representations do not necessarily contain the same number of joints or joint placements within the human body. An online approach must be bi-directional stable, meaning multiple retargeting steps without any content change should not create "ghost" movements, jitter, or drift.

The ideal retargeting function for applying neural-network-based motion synthesis has the following requirements.

1. **Real-time retargeting**: animation is generated frame-by-frame and thus needs to be retargeted to the visualized character in real-time.
2. **No manual post-processing**: as the frames are produced directly for the given environment and state, there is no possibility for manual post-processing after the development time.
3. **Manual adjustment**: the retargeting function needs to be adjustable by a human to increase realism and to match the motion to individual characters.

By defining an intermediate or internal skeleton representation and mapping each avatar against this representation, it is possible to transfer the animation from one to another skeleton [Mon+00; Fen+14]. When the animation is transferred to a character of a different scale, the end effectors are not necessarily in the right position anymore. For example, transferring the motion from a larger to a smaller character creates foot-skating artifacts during locomotion or hand misplacements during reach animations. To adjust the animation to a differently scaled avatar, existing approaches rely on additional annotation [Fen+14], semantic information [Hec+08], or direct access to the animation [Mon+00] to apply IK during the animation to ensure, that the relevant end effectors reach the correct position after retargeting. Without annotations, there is no universally correct way

of applying IK: In case of reaching, there should be an IK constraint on the reaching hand when it touches the target. However, in case of locomotion, the hands should not have IK constraints, but the global translation needs to be scaled down to avoid foot skating.

We utilize an intermediate transfer skeleton to retarget motion between an arbitrary number of skeleton representations. For each skeleton representation, two retargeting methods are required: retargeting from the target representation to the intermediate representation (*RetargetToIntermediate*, *R2I*) and from the intermediate representation to the target representation (*RetargetToTarget*, *R2T*). The methods need to be inverse functions to ensure bi-directional stability: applying R2I and R2T in succession should always result in the original pose ($R2T(R2I(pose)) = pose$). As a result, the retargeting function is stable, and successive application of the functions without changing the pose should, in turn, not affect the pose.

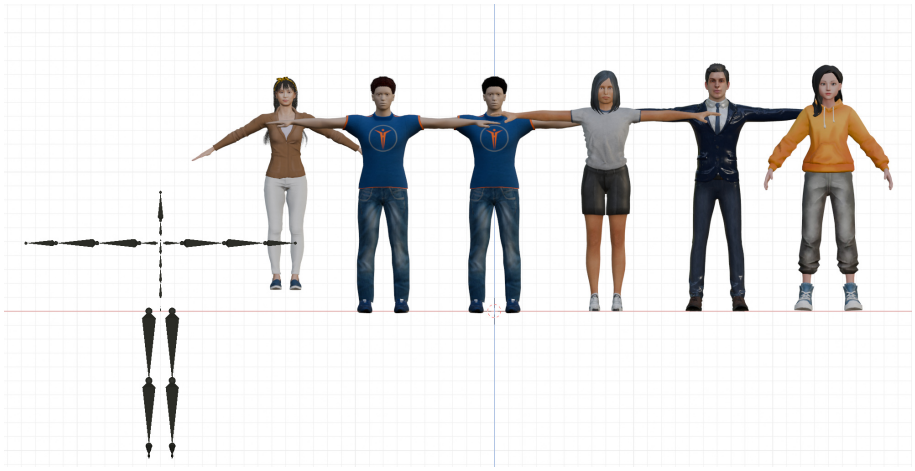
In this section, a numeric approach for motion retargeting with a function that can be fine-tuned during development time is presented. It is bidirectionally stable and fast, requiring only a single reference pose as a configuration. In addition, experiments on an extension to enable motion transfer between differently-sized characters are presented.

5.2.1 Skeletal Representation

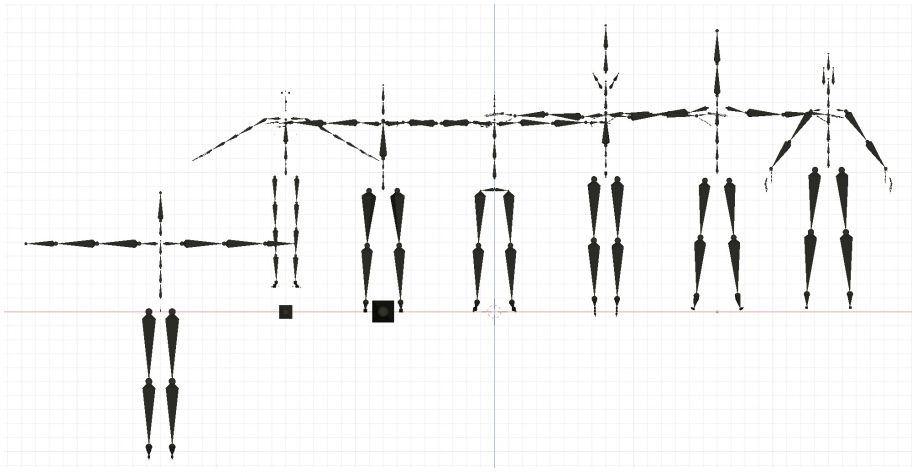
Skeletons are usually considered as a kinematic chain of connected joints. However, skeletons can differ in size, hierarchical description, and zero posture. Particularly when two skeleton structures have a different number of joints, there is no unique retargeting function, and additional optimization (e.g., inverse kinematics with multiple targets) is required. During real-time animation transfer, this results in temporal instability, resulting in jitter and drift of animation after multiple retargeting steps in succession, which was confirmed in experiments.

Certain similarities and differences are visible when analyzing different humanoid skeleton representations, as shown in Figure 5.1. Arms and legs are usually defined with three joints each (hip, knee, ankle; shoulder, elbow, wrist) and can contain additional joints to connect to the spine (clavicle, left/right pelvis). The spine and neck complex are differently modeled in different skeletal representations. The natural curve, as seen in Figure 5.2b, is not always followed accurately and if, with a different number of joints and levels of detail. The hand is sometimes abstracted to a single joint, not allowing articulated finger movements. While the human hand, as shown in Figure 5.2a, has a high number of carpal and metacarpal bones within the palm, virtual characters usually animate the fingers with three joints: between the metacarpal and proximal phalange bones, between the proximal and intermediate phalange bones and between the intermediate and distal phalange bones. Due to the thumb not having an intermediate phalange bone, the first joint is already between the carpal and metacarpal bones.

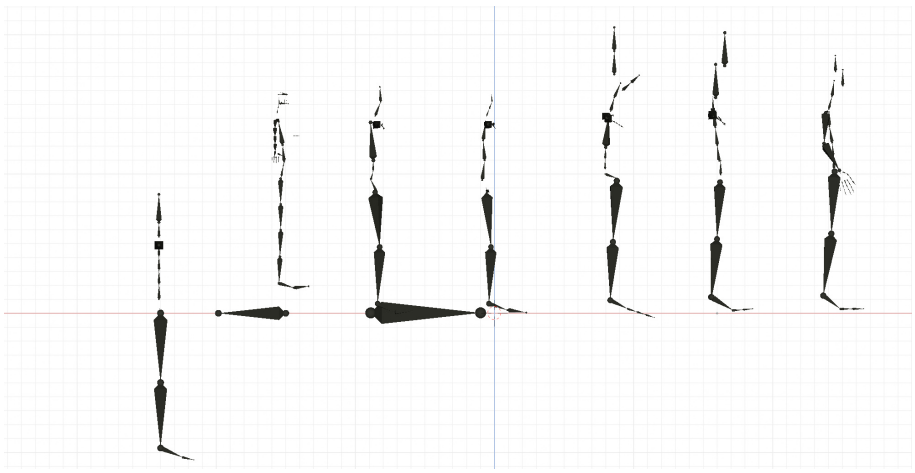
We violate the assumption of kinematic connections between the joints to overcome the temporal instability due to inconsistencies in motion transfer. Additional degrees of freedom are added to the starting joint of each sequence (e.g., left and right shoulder, left and right upper leg, metacar-



(a) Rendered Characters



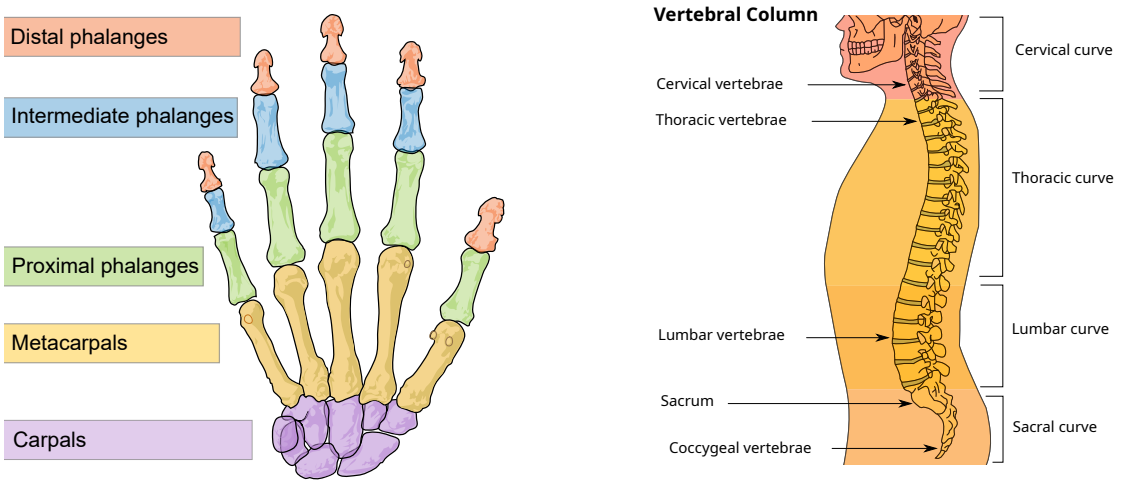
(b) Front View



(c) Side View

Figure 5.1: Different skeletons for different character models shown in an orthographic projection. From the left: Xsens, iClone[a], MakeHuman/CMU, MakeHuman/GameEngine, Valid [Do+23], Mixamo[b], Ready Player Me.[c] All models are in their zero posture and positioned at their zero height in Blender. The red line denotes the world zero height.

[a] `https://www.reallusion.com/de/iclone/`
[b] `https://www.mixamo.com/`
[c] `https://readyplayer.me/de`



(a) Bone structure of the hand

(b) Bone structure of the spine

Figure 5.2: Structure of the real human hand and spine complex. Public domain figures provided by the Wikimedia Foundation.

pophalangeal joints) and each spine joint. While this technically results in a non-kinematic skeleton, the visualization of humanoid characters usually has a mesh that the skeleton transforms. Minor translations of the joints within the local neighborhood do not reduce the visual quality of the motion. In case of the palm, it can even increase the motion complexity without increasing the complexity of its representation. As a result, we propose an intermediate skeleton representation with a pragmatic selection of joints and translational freedoms as depicted in Figure 5.3. The skeleton is described in an extended BVH (Bio-Vision Hierarchy) format[47], where each joint contains the following information: ID (string), type (enum), base position (Vector3) in parents coordinate system, base rotation (Quaternion) in parents coordinate system, parent (ID), channels (either WXYZ rotation only or XYZ translation and WXYZ rotation).

The BVH format does not consider a base rotation, which results in each joint having the same initial rotation as the world coordinate system in its base configuration. As a result, rotations around one axis are not necessarily related to a meaningful rotation of the bone. If there is a base rotation in each joint, the joint can be pre-aligned with the corresponding bone, such that, for example, the rotation of the x-axis of the hip joint results in the upper leg bending and not twisting. The base rotations and offset directions of the intermediate skeleton are well-defined up to the scale of the simulated human. They can be found in the Appendix A.2.1. The animation data is represented as a list of floating values corresponding to the channels of the joints in depth-first order.

[47] `https://research.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html`

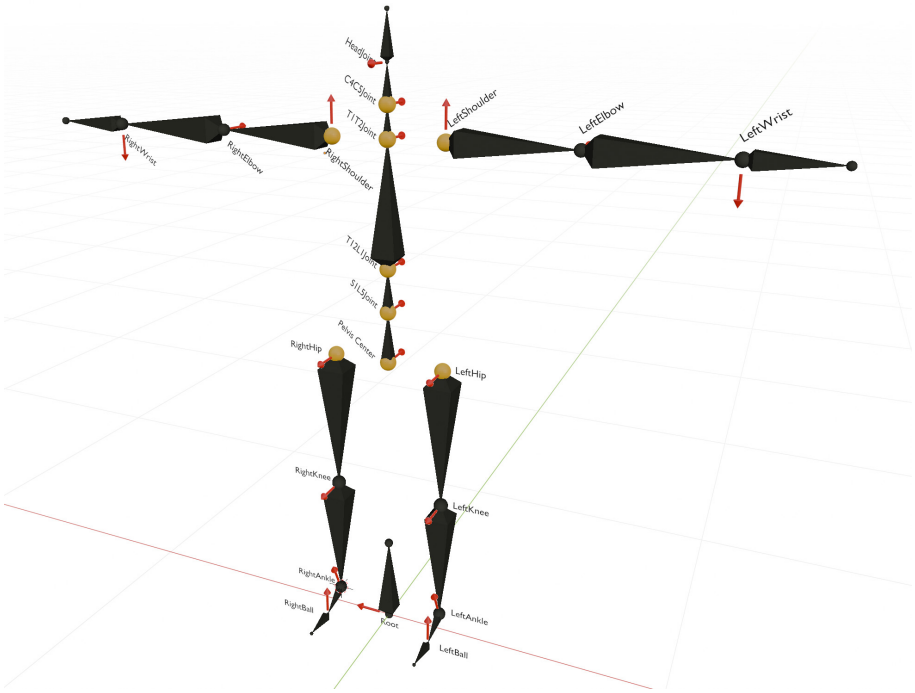


Figure 5.3: Visualization of the intermediate skeleton representation. Joints with translational degrees of freedom are highlighted with a yellow sphere. The Y-axis points along the bone (black), and the x-axis is along the primary interaction axis (red). The finger joints are not included for easier visualization.

5.2.2 Retargeting Method

The retargeting method requires a single reference pose and a joint map for the configuration. The pose should display the character in an upright standing position, facing straight ahead, with legs parallel and orthogonal to the ground and arms straight and parallel to the ground (a T-pose). The fingers should be on a flat plane parallel to the ground, with the thumbs at a 45° angle to the index finger. Figure 5.4 displays an example of this reference posture. The joint map defines the correspondence between character joints and joints of the intermediate transfer skeleton. Figure 5.5 displays the retargeting map corresponding to a reference posture.

Based on the reference pose, the intermediate skeleton is automatically scaled to the same size as the character skeleton, and the difference in rotation (offset rotation) and position (offset position) are computed. We define R_t, P_t as the rotation and position of a joint in the reference pose of the target character and R_i, P_i as the rotation and position of the corresponding intermediate joint. The positional offset in the target joint space O_t and intermediate joint space O_i can be computed as

$$O_t = R_t^{-1} * (P_i - P_t)$$
$$O_i = R_i^{-1} * (P_t - P_i)$$

For a single pose, the joint position p_t and rotations of the target joint r_t can be retargeted to the intermediate joint position p_i and rotation r_i with the following functions

$$r_i = r_t * R_t^{-1} * R_i$$
$$p_i = p_t + r_t * O_t$$

and vice-versa in the opposite direction with

$$r_t = r_i * R_i^{-1} * R_t$$
$$p_t = p_i + r_i * O_i$$

Thus, equality under subsequent retargeting and inverse retargeting is maintained. A proof of this equality is shown in the Appendix A.3.1. While floating point arithmetic is an imprecise approximation of the mathematical accuracy, it does not introduce instability to subsequent retargeting in practice. The mean joint position error was in a submillimeter range even after over 1000 subsequent retargeting steps of the same pose.

5.2.3 Retargeting Configuration

The configuration of the proposed retargeting approach is of utmost importance, to improve the naturalness of the final visualization. A configuration tool was developed to enable an easy and adjustable definition of the reference pose. Inside of the tool, the impact of the reference pose can be directly visualized, adjusted, and refined. The configuration approach follows the following three steps:

1. **Global Positioning**: positioning and rotation of the avatar in an upright position
2. **Joint Mapping**: mapping of joints between the avatar and the intermediate skeleton
3. **Pose Alignment**: posing the avatar in the reference pose

The retargeting configuration tool is published on GitHub, including its documentation.[48]

Global Positioning. Avatars must be upright standing on the ground. In the first step, the root and pelvis transform (or similar) can be selected, and the global position and rotation can be manually adjusted. To make this process more intuitive, the camera is placed in a fixed location and aimed to make the avatar look into the camera (see Figure 5.4).

Joint Mapping. Although there have been some approaches for automatic joint mapping [San+13], our experiments did not show a solution with sufficient robustness to operate for all kinds of skeletons. As the work is restricted to humanoid skeletons, a heuristic similar to Feng et al. [Fen+14] is developed to create an initial map of the avatar's joints to the intermediate skeleton. In particular, the pre-alignment step is utilized to detect left and right body sides. Legs and arm segments are

[48] https://github.com/dfki-asr/MOSIM-Unity/tree/master/Tools/SkeletonConfigurator

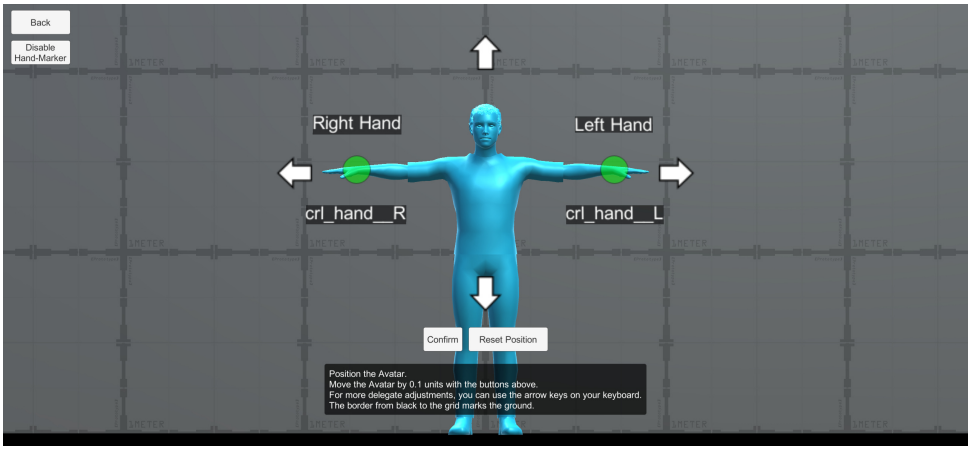


Figure 5.4: Initial alignment of a character model in the retargeting configurator. The user can adjust the automatic pre-alignment to rotate the character into a forward-looking position with the feet on the ground, as not all representations contain a character in this base position.

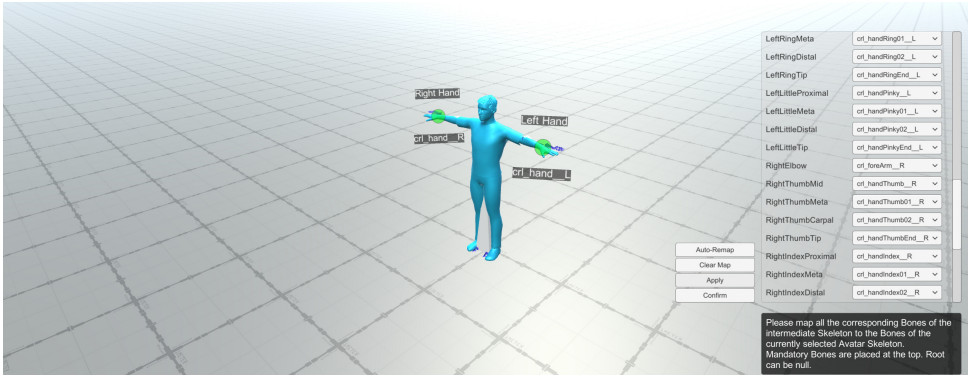


Figure 5.5: After initial alignment, a semi-automatic joint mapping is used to map the joints of the target character to the intermediate skeleton. The user can adjust the joint mapping with the dropdown options on the right.

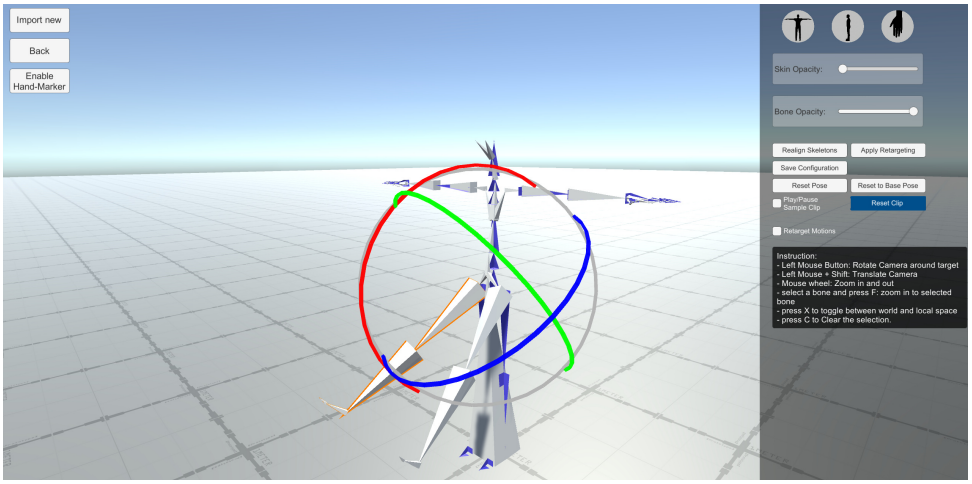


Figure 5.6: The target character pose (gray skeleton) can be manually adjusted to match the intermediate skeleton's reference pose (blue skeleton) after finalizing the joint mapping. Transformation on opposing joints (e.g., left and right upper leg) are mirrored for easier adjustment.

considered to have at least three body segments (e.g., shoulder, elbow, wrist). The approach works without any requirements for joint naming conventions but assumes a pose similar to a T-pose. In addition, the approach fails for artificial joints, e.g., accessory joints (e.g., hat, cloth, hair, chest, tail, etc.) and bone twist joints. The user can adjust the map to compensate for failures of the heuristic (see Figure 5.5).

Semi-Automatic Pose Alignment. While an exact alignment of the mapped joints leads to a mathematical exact positioning of the joints, it does not necessarily result in the most natural transfer of motion. As the placement of joints within the mesh is up to the designer's discretion, the exact position of joints between two different avatars might be dissimilar. In addition, motion capture data can contain bias due to miscalibration, uncommon joint placement, or unconventional movements (e.g., raised shoulders, hypermobility, bowlegs, etc.). As described above, spine complexes already differ significantly and often require manual adjustments. Adjusting the shoulder and elbow positions and the hip and knee positions can increase the subjective naturalness of the retargeted motion. Thus, a semi-automatic alignment process was implemented. The user can manually execute the automatic alignment and manipulate the reference pose (see Figure 5.6). Automatic alignment maintains manual changes and thus can be repeated to realign joints lower in the hierarchy.

Intermediate Skeleton Scaling. The intermediate skeleton representation is defined by the number of joints and their base rotations up to the scale. It is scaled to fit the target avatar after defining the joint mapping and aligning the avatar in the base position. This ensures that the intermediate skeleton closely matches the size of the target avatar. As the animation data (rotations, offsets) are defined within the local joint coordinate systems, animation can be copied from one intermediate skeleton to the other, maintaining the general style and look. However, in case of different sizes between the source of the motion and the visualized avatar, end-effector positions are not necessarily at the same location. As there is no trivial way to correct this scaling error without additional semantic knowledge about the motion, motion models are required to provide animation that matches the size of the target character.

5.2.4 Evaluation

Performance

The computational performance of retargeting a pose from one meshed character representation to another via the intermediate skeleton is evaluated. 20,000 retargeting steps are computed to calculate the average computation time. The evaluation is executed on a gaming PC (AMD Ryzen 9 16 Core CPU, NVIDIA RTX 3080 GPU, 64 GB RAM). The retargeting algorithm is implemented in C#. It can be deployed either as a service using the Apache Thrift[49] framework for remote procedure calls or as a native service within the Unity game engine. For both deployments, the following steps are evaluated.

[49] `https://thrift.apache.org/`

Table 5.1: Runtime performance of remote and local retargeting services. The core implementation is the same. Local retargeting utilizes the retargeting code directly within the Unity game engine, while remote retargeting utilizes the deployed Thrift service via a local network connection. The computation duration without the overhead of serialization and the localhost network connection is shown in brackets.

	Remote Retargeting	**Local Retargeting**
Reading Posture	0.016 ms	0.014 ms
Source - Intermediate	0.487 (0.06) ms	0.05 ms
Intermediate - Target	0.672 (0.06) ms	0.09 ms
Apply Posture	0.03 ms	0.059 ms
Total Time	1.204 ms	0.212 ms

1. **Reading Posture**: Computation of global joint positions and rotations of the target character
2. **Source to Intermediate**: Retargeting from the source to the intermediate transfer skeleton
3. **Intermediate to Target**: Retargeting from the intermediate skeleton to the target skeleton
4. **Apply Posture**: Applying the new posture to the target character

All steps are executed in succession during a single game engine update. Using the local implementation, retargeting requires 0.212 ms in total. Using the remote retargeting service, the pose is serialized and sent to the service via the loopback interface to be retargeted either from the source to the intermediate or from the intermediate to the target representation. In this test, the service performs both steps separately on purpose. The detailed results shown in Table 5.1 indicate that serialization by thrift and the TCP data transfer creates significant overhead, resulting in 1.204 ms for the remote retargeting. If required, further optimization could be achieved by merging both retargeting steps in a single service call. In both cases, however, the computation time is sufficient for the application in pedestrian or manufacturing simulations. It can support real-time animations while providing flexibility for a service that does not need to be implemented. This is useful for other engines – like Unreal Engine – as the service can be utilized without requiring a re-implementation in C++.

Accuracy

The retargeting approach is stable and can be repeatedly called, as it fulfills the identity property of $R2T(R2I(pose)) = pose$. However, the current implementation has a major source of inaccuracy by design. As not all joints have translational channels, inaccuracies in the calibration pose can lead to slight misplacements of the end-effectors, particularly in the wrist and feet. In practice, however, this misplacement was only minor and barely visible. Thus, a numeric evaluation of the retargeting accuracy between different hierarchies is irrelevant due to the intrinsic inaccuracy of the design.

Generalization

We tested the generalization with different characters and skeletons. While simple kinematic chains like MakeHuman and Mixamo are unproblematic, more advanced skeletons like the ones generated with the iClone character generator are more challenging. These rigs contain different auxiliary joints, controlling the twist of the bone and allowing a more fine-grained transformation of the mesh. As these twist-bones are not supported, the automated procedure to find an initial mapping fails frequently, requiring a developer to define the bone map manually. We extended the configurator to load BVH animations generated by motion-capturing or data-driven neural networks. One of the challenges of BVH data is the unavailability of a reference mesh and missing rotation data of the rest pose. Thus, joint placements with a mesh and the joint twist can not be trivially derived. Several manual adjustments to the automated alignment are necessary to achieve a plausible look on a meshed avatar.

5.2.5 Extension Towards Multi-Scale Retargeting

Scale is not considered when transferring the motion between two characters in the approach described above because the semantics of the motion are not known. If, for example, the motion is reaching a target point, the absolute hand position in space should match exactly between differently scaled avatars (see Figure 5.7). During walking, however, moving the hands up or down to match the absolute position would result in an unnatural pose (e.g., "chicken walk"). Inverse kinematics can be utilized to adjust the animation procedurally. However, it can create additional complexity and does not generalize to arbitrary actions.

With Jan Bohnerth, we investigated a different approach for multi-scale retargeting in his Bachelor thesis [Boh21]. When copying the animation data between two differently scaled skeletons of the same hierarchy, the motion no longer fulfills the input constraints. A taller person walks further than a smaller person, and a smaller person would reach further below than a taller person. Figure 5.9 displays an example of the animation data being copied from the source character on the left to the target character on the right. Subsequently, this mode of animation transfer is called direct transfer or direct copying of animation data.

In our approach, instead of adjusting the motion scale during retargeting, the input constraints during motion generation are rescaled to achieve the correct result. In the example of reaching in Figure 5.7, the target position for the reach motion needs to be moved higher for the animation model animating the motion source character for the target character to reach the correct position (blue ball) after direct motion transfer. Figure 5.8 displays this general concept. We evaluated two different approaches to implement this concept. The first approach utilizes a database of input constraints and directly transferred constraints. The second approach adjusts the implementation of the motion generator to leverage the knowledge and functionality available within the motion generator itself.

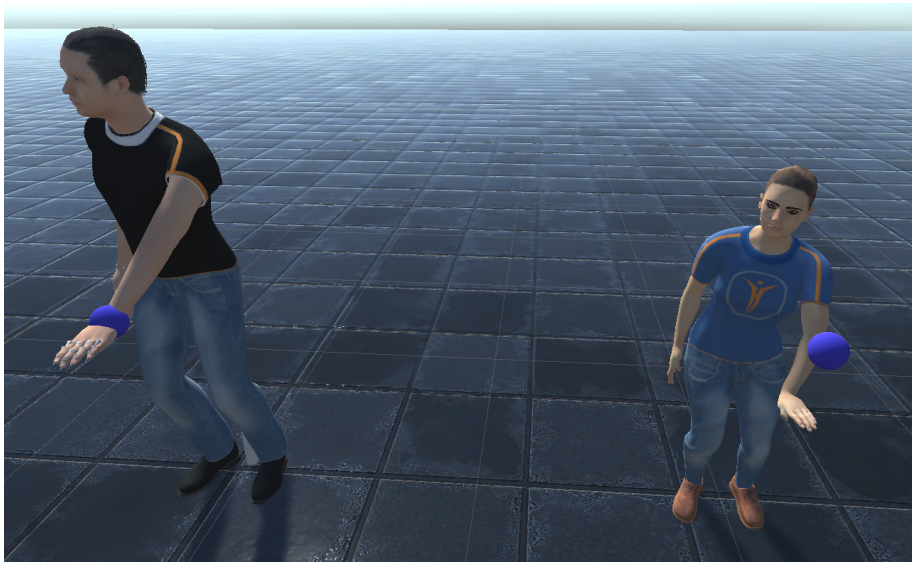


Figure 5.7: The direct transfer of motion between the large (left) and small (right) character will result in the smaller character not reaching the target position (blue ball). (Figure originally from [Boh21])

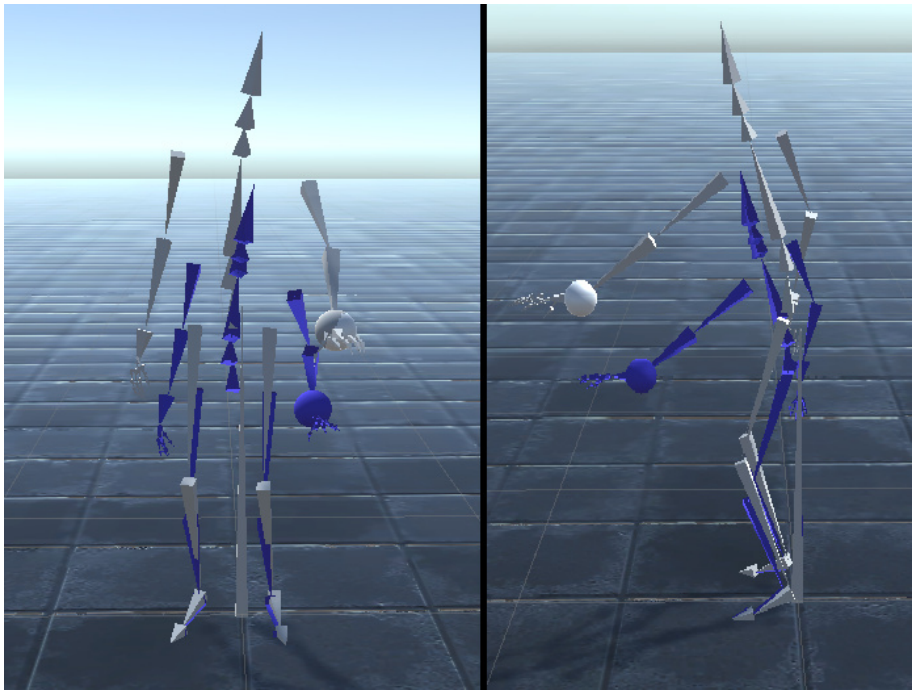


Figure 5.8: Front (left) and side view (right) of the motion source character (gray) and the target character (blue). The target position (blue sphere) was modified for the motion character (gray sphere) in such a way, that the target character reaches the target position after direct motion transfer. (Figure originally from [Boh21])

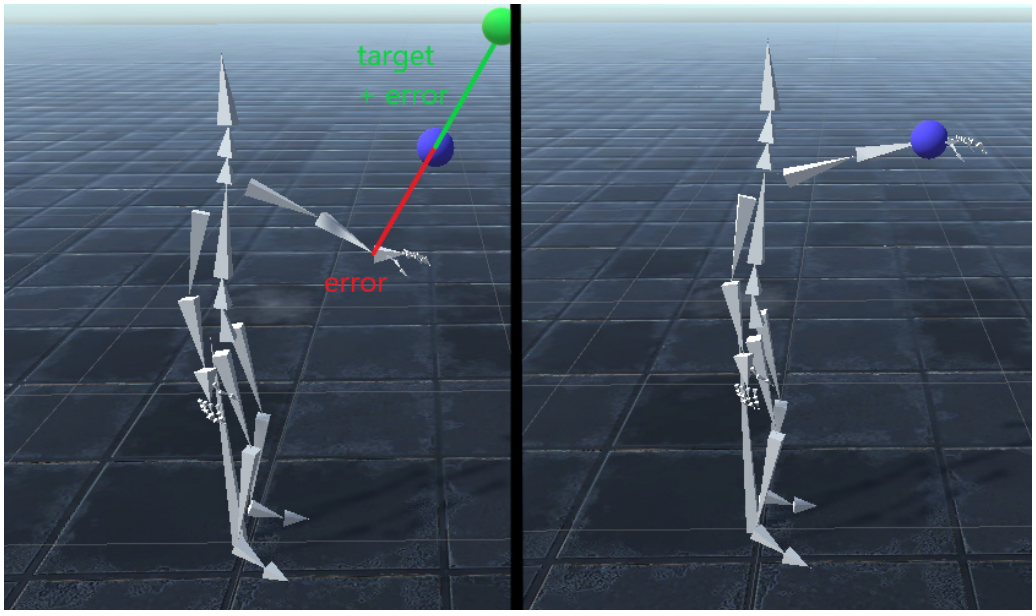


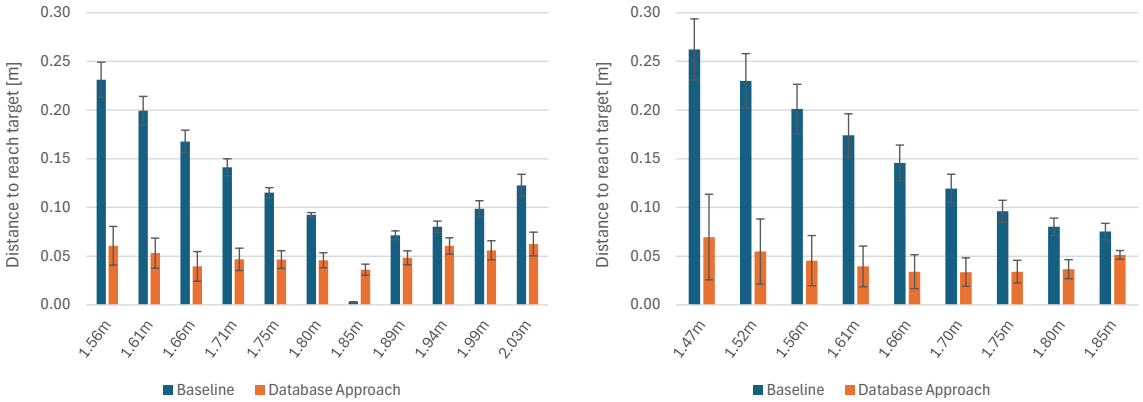
Figure 5.9: The displacement error of the directly copied pose (gray skeleton) to the goal position (blue ball) is computed, and that target position is shifted by this difference. (Figure originally from [Boh21])

Extended Approach for Multi-Scale Retargeting

The two approaches are implemented in the example of a reach motion, animating the character to reach for an object. Following the norm ISO/TR 7250-2:2010 of basic human body measurements for technological design [Int10], 20 characters within the distribution for male and female size proportions were created using the MakeHuman character creator. For the database, 7,560 reach points are sampled in front of the character, and a motion generator simulating a reaching motion is utilized to compute reach animations for these positions. The animations are transferred using direct animation copying to each of the target characters. For each data point, the size of the target skeleton, the original target point, and the target point after transfer are stored (total size of 5.3 MB). The database is accessed using the skeleton size and the target point after transfer.

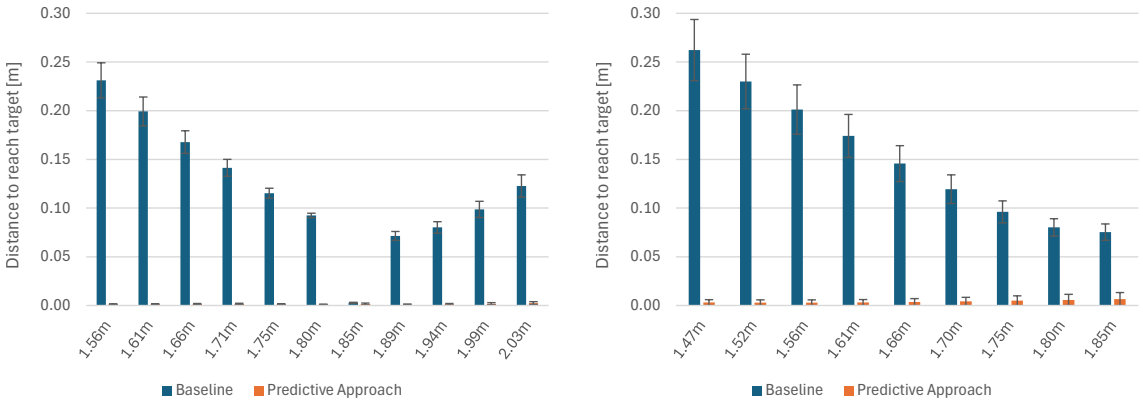
In the **database approach**, the database is used during runtime to select the three nearest neighbors (approx. 0.015 ms per lookup) to the target point after transfer for the respective skeleton size. The blended position of the three neighbors is utilized in the instruction to the motion generator, and the resulting joint rotations are then directly copied to the target skeleton. This approach requires no additional knowledge about the motion generator and uses it as a black box.

In the **predictive approach**, access to the internal functions of the motion generator is required, and the simulation code needs to be adjusted. When a new target constraint is provided to the animation generator, the final reach pose is simulated first, and the animation pose is directly copied to the target character. The displacement of the goal position and the wrist in the transferred and thus scaled pose is evaluated. The constraint utilized for animation generation is then moved by the negative displacement vector, and the whole animation is generated as usual. Figure 5.9 shows the concept of this approach.



(a) Male Characters **(b)** Female Characters

Figure 5.10: Accuracy after retargeting with the database approach for each character scale. The baseline is computed by direct copying of joint rotations. Error bars denote one standard deviation. (cf. [Boh21])



(a) Male Characters **(b)** Female Characters

Figure 5.11: Accuracy after retargeting with the predictive approach for each character scale. The baseline is computed by direct copying of joint rotations. Error bars denote one standard deviation. (cf. [Boh21])

Evaluation

The motion for the evaluation is generated with a reference character (male, 1.85 m height). 3,182 reach points are sampled and manipulated for each target character. Hence, the input to the motion generator always depends on the target character to which it is transferred. The error is measured as the distance of the wrist to the reach target constraint.

Both methods improve the accuracy compared to the baseline. Figure 5.10 shows the results for the database approach. While the baseline error increases with an increasing difference with respect to the motion generation character, the error remains almost constant over all different body sizes at around 5 cm. The results using the predictive approach are shown in Figure 5.11. Unlike the database approach, the predictive approach results are almost perfect, with an average accuracy below 1 cm.

The database approach does not require access to the motion generator's internal structure and thus maintains its black-box character. However, the predictive approach demonstrates how developers can incorporate scale information already at the development stage to perfectly adjust to the target character scale.

5.2.6 Discussion

We have presented a fast, robust, real-time solution for retargeting motion between skeletal hierarchies. It uses numerical computations with a human-defined reference posture to transfer motion and produces consistent results among skeletal hierarchies. An extension towards scale-independent retargeting was shown. While retargeting between different apriori unknown skeletal hierarchies is feasible with a general solution, the motion developer should already perform the scale adjustment in the development stage.

Our solution fulfills the requirements to retarget motion generated by neural networks, but it is still imperfect. The intermediate transfer skeleton is a practical approximation but is not biomechanically accurate. The spine, however, is not approximated well. A plausible visualization after retargeting can be achieved by violating the connectivity of joints. However, a better internal representation could improve the retargeting's quality, explainability, and accuracy.

In the presented approach, some joints have additional degrees of freedom (e.g., in the spine), while others do not (e.g., in the knee and elbow). While limiting the translational degrees of freedom of joints ensures a consistent length of the connecting segments, it can result in minor misplacements of the end effectors during retargeting. Hence, we propose that all joints should be allowed additional translational freedom. By allowing a little movement inside the joints, the intrinsic inaccuracy of the current model would be removed. Having translational freedom effectively scales the bones of the meshed characters. While minor scaling will not be visible, larger scaling of bones should be avoided. Additional optimization for the visualization (e.g., inverse kinematics) can further improve the visual result. Extra care, however, is required to ensure that the identity property of the retargeting function is maintained.

As outlined in Section 5.2.5, retargeting between different scaled characters is nontrivial and depends on the semantics of the motion. Two approaches are presented for scale adjustments, which work reasonably well for simple animation models. However, more complex animation systems will require more complex heuristics and pre-calculations. Combining both approaches would be beneficial. With simple animation scaling, an animation database can be created during development time by directly copying the animations from larger to smaller skeletons and vice versa. With additional optimization, scaling artifacts (e.g., self-collision, floating, foot-skating) can be compensated. Using the database, a scale-dependent motion model can be created. By adding the character scale as a condition for a cVAE or diffusion model, for example, the model can learn to simulate the actions for the correct character scale from the start.

5.3 The MOSIM Framework: Distribution of Motion Simulation and Transfer to External Target Engines

Character animation systems are usually built monolithical, i.e., a single enclosed system, forcing all animation methods to comply with their procedures. Novel animation techniques, such as neural networks, deep learning, and advanced computational methods for kinematic simulations, do not necessarily comply with the native animation system and, thus, are not integrated trivially. In addition, they are not necessarily written in the same programming language or they even require their own eco-system to be executable, including additional license fees or the transformation of the existing pipeline to a new software stack. On the other side, domain engineers who create simulations for autonomous driving scenarios want to utilize modern animation systems without having the technical expertise to implement and integrate them, as they are no animation experts. They require a modular system that is controllable by action-specific instructions (e.g., walk to a position, sit down, etc.).

In simulations for autonomous vehicle development and Industry 4.0 applications, the modularization of digital twins has received significant attention in the past years. Concepts like the Functional Mockup Interface (FMI) and resulting Functional Mockup Units (FMU) [Blo+11] as well as the Open Platform Communications Unified Architecture (OPC-UA) [EM11] aim to enable the modularization of technical simulation modules, for example for the motor, the transmission system or the camera. Particularly for preserving intellectual property, simulation models are closed-source black boxes that provide a partial simulation for an individual component. Combining the components in a co-simulation allows more complex digital twins to be simulated.

Both of these frameworks, however, are not suitable for the modularization of human digital twins or their animation system as they focus solely on the integration of dynamic linked library (DLL) of C-libraries, limit communication formats (mostly floating values and lists of floating values), and have limited data sharing capabilities. To simulate virtual characters in different environments, environment information (e.g., positions, rotations, bounding boxes, etc.) must be shared. Combining such animation modules requires more structured information on the skeleton hierarchy, the animation data, and the environment.

To overcome the aforementioned challenges, we present the **MOSIM Simulation Framework**, a distributed modular framework for simulating human agent motions and behavior. The framework modularizes action simulation in separate modular motion units (MMUs), provides the basis to exchange scene information, and the ability to configure and constrain motion instructions in a distributed system. It was integrated into several 3D simulation environments, including Unity, Unreal Engine, and Blender. The whole implementation is published as an open-source solution on GitHub[50].

[50] `https://github.com/dfki-asr/MOSIM`

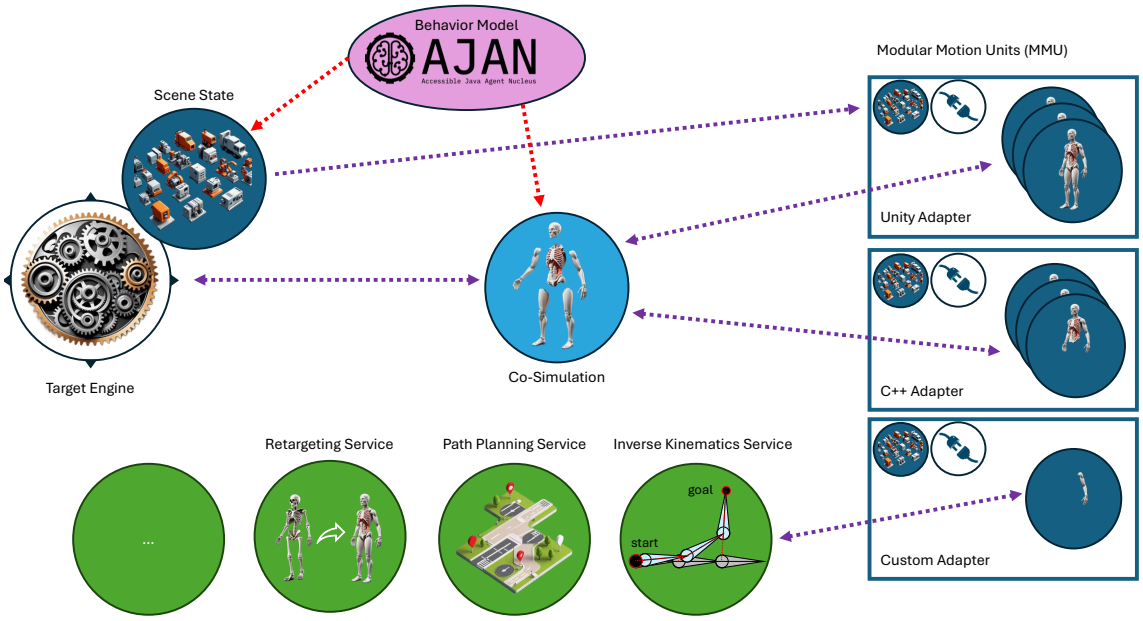


Figure 5.12: Overview of the MOSIM Framework. The target engine on the left is the primary simulation environment, hosting the scene state. The co-simulation in the middle combines motions generated within the MMUs on the right. The MMUs utilize the services on the bottom if required. Each component (co-simulation, services, target engine) runs in its executable environment. The MMUs are combined in language-specific adapters and generate either full-body animations or only partial animations. Purple connections display high-frequent communication channels executed in each frame, while red connections denote low-frequent communication channels.[51]

5.3.1 System Overview

The MOSIM Framework is a distributed simulation framework that communicates over remote function calls. This allows different simulation components to be executed in different software environments, reusing existing tools such as Unity or Blender. MOSIM utilizes Apache Thrift[52] as its backbone communication method. Access to the individual components is defined through their interface definitions in Thrift. An overview of the system can be seen in Figure 5.12. The following section provides a high-level overview of the framework, followed by a more detailed explanation of each component.

The **target engine** is the main component that an end-user is interested in. It displays the animations and provides the central tick, synchronizing the simulated time of all components. In addition, the target engine hosts the scene. It provides information on objects inside the scene and their positions, accessible through the scene access service. The native scene graph of the target engine requires some augmentation to enable synchronization, and a MOSIM endpoint must be added to communicate with the MOSIM Framework. Most of the communication with the framework is performed by querying a new pose of the co-simulator for a given elapsed simulation time and

[51] Icons are based on generations from `hotpot.ai/logo-generator`
[52] `https://thrift.apache.org/`

synchronizing the scene state with the other components. The **co-simulator** combines the execution and output of several modular motion units. It merges motion generated by different models into a single agent representation. The motion is controlled by sending **instructions** to the co-simulator, which is possible through the co-simulation access. A **modular motion unit (MMU)** is simulating the motion for a specific task, like walking to a location, reaching for an object, or waving in a direction. Thus, animations generated via MOSIM are not replayed animation sequences but an actual simulation of a goal-directed movement. The MMUs and the co-simulator share the same interface; hence, stacking multiple co-simulators is possible. MMUs are hosted within a language or environment-specific **adapter**, which reduces the number of required open network ports and performs central tasks of buffering the scene access and streamlining the network communication. Combining a single MMU with a single adapter in the same executable program is possible if required. Certain tasks, such as inverse kinematics, retargeting, and path planning, are required by multiple components. Consequently, they have been packed into standalone **services** to enable reusability. A motion exchange skeleton structure is proposed to combine the simulation results of MMUs of different providers. An advanced **behavior model**, for example, AJAN [Ant+19] (Advanced Java Agent Nucleus), controls the agents and adapts to the action sequences depending on the environment state.

During the simulation, the target engine calls the `DoStep` method of the co-simulator to receive new poses. The co-simulator evaluates which MMU is currently active and calls the `DoStep` method of the respective MMU to generate the new frame pose and merge poses of multiple MMUs to a consistent output. The MMU utilizes the retargeting service to transform the MOSIM representation to its internal character skeleton and may utilize the inverse kinematics service to compute an IK solution. The behavior model uses the `AssignInstruction` method of the co-simulator to assign new motion instructions for the required actions to be performed. In turn, the co-simulator will distribute the instruction to the relevant MMUs and orchestrate them.

The following section will provide a more detailed overview of the different components and their interfaces.

5.3.2 MOSIM Components in Detail

Modular Motion Units (MMUs)

Modular motion units (MMUs) encapsulate a motion synthesis methodology as a stateful service. An MMU generates motion for a task defined by goals and parameters, which are provided via an `MInstruction` (see Listing 5.14). The MMU is responsible for synthesizing the motion to reach this goal and adjusting this motion for the specific character model. The implementations of MMUs are generating animations for walking to a position, moving an object, picking up objects, drilling a hole, or climbing a ladder.

The interface with its most important functions is shown in Listing 5.2. It describes three fundamental methods: `Initialize`, `AssignInstruction`, and `DoStep`. `Initialize` receives an

Listing 5.2: Most important functions of the `MotionModelUnit` interface. The full interface can be found in our Github repository.

```
service MotionModelUnit
{
    MBoolResponse Initialize(1: MAvatarDescription avatarDescription,
        2: map<string,string> properties),
    MBoolResponse AssignInstruction(1: MInstruction motionInstruction,
        2:MSimulationState simulationState),
    MSimulationResult DoStep(1: double time, 2: MSimulationState simulationState),
        list<MConstraint> GetBoundaryConstraints(1:MInstruction instruction),
    MBoolResponse CheckPrerequisites(1:MInstruction instruction),
    MBoolResponse Abort(1:string instructionId),
    ...
}
```

Listing 5.3: Excerpt of the definition of the `MMUDescription` data structure.

```
struct MMUDescription
{
     1: required string Name;
     2: required string ID;
     3: required string AssemblyName;
     4: required string MotionType;
     6: required string Language;
     7: required string Author;
     8: required string Version;
     9: optional list<MConstraint> Prerequisites;
    11: optional map<string,string> Properties;
    12: optional list<MDependency> Dependencies;
    13: optional list<string> Events;
    14: optional list<MParameter> SceneParameters;
    15: optional list<MParameter> Parameters;
    ...
}
```

avatar description and is utilized to initialize the system, including all internal steps to adjust to the given skeleton size. `AssignInstruction` receives a simulation state and a motion instruction to be performed by the MMU. The motion instruction contains information about the goal constraint, boundary constraints, and additional parameters the MMU requires (e.g., which hand to use, how fast to walk, etc.). During the simulation, the `DoStep` method is called. It receives the current simulation state and a time step and is supposed to simulate the next pose given the current system state and time step. This function is similar to the methods called during the update of the game loop in various gaming engines.

Additional methods in the MMU interface allow for better planning and sequencing. The function `GetBoundaryConstraints` returns a list of constraints required for a seamless transition while `CheckPrerequisites` checks whether the execution of a certain instruction can be started.

For a machine-readable description of an MMU, the data structure `MMUDescription` is conceived (Listing 5.3). In addition to parameters for identification (name, ID, author, etc.), it contains information on the prerequisites, properties, parameters, and dependencies of other components like services. The `MotionType` is the semantic identifier of the action. It is a concatenation of the module's class, the action, and optional details. Three module classes are utilized:

1. Locomotion: Any MMU changing the global position and rotation of the agent significantly
2. Object: Any MMU interacting and manipulating objects in the environment
3. Pose: Any MMU animating the character without environment interaction

Examples of motion types are “Pose/Idle” for playing an idle animation, “Locomotion/Walk” and “Locomotion/Run” for walking and running modules, and “Object/Retrieve” and “Object/Place” for retrieving and placing objects.

Co-Simulation

The co-simulator invokes, combines, and homogenizes the different MMUs. Its interface is the same as the MMU; thus, a co-simulator may consist of additional co-simulators as well. It is a module within the MOSIM Framework and can be replaced by a custom implementation. The published reference implementation utilizes a hierarchical execution scheme that calls less specific motions (e.g., full body motions) first and uses their result to invoke more specific motions (e.g., finger motions) second. The specificity of the MMU is defined by its motion type. In addition, the co-simulation utilizes the inverse kinematics service to ensure constraints set by the individual MMUs (e.g., keeping the hand at a specific global location). A more detailed explanation of the functionality and implementation of the co-simulator can be found within Gaisbauer et al. [Gai+18] and Gaisbauer [Gai22].

Unlike an animator, the co-simulator is designed to combine the motions of different black-box MMUs during runtime. Thus, it has no precise knowledge about the future except for the goals defined for executing each task. As a result, typical optimizations, such as blending, temporal alignment, and pre-emptive IK solving, are not solved by the co-simulator but must be considered by the individual MMUs. Reactive IK solving to maintain joint constraints continues to be feasible and is integrated into the co-simulator.

Motion Representation and Transfer

Different animation developers and even different MMUs of a single developer can consider different representations (e.g., skeletons) of the virtual agent to be animated. To transfer and combine the motion of different MMUs, a common understanding of a virtual character must be established through an exchange skeleton.

The exchange skeleton is a semi-kinematic structure using an extended BVH format[53] (Bio-Vision Hierarchy) with additional offset rotations in the zero posture. Listing 5.4 shows the data structures utilized for posture (`MAvatarPosture`) and joint (`MJoint`) definition. Each joint is represented by a position (`MVector3`) and a rotation (`MQuaternion`) and contains the IDs of its parent joint and the reference joint in the intermediate skeleton (`MJointType`). The position and rotation are provided in the coordinate system of the parent joint (joint local coordinate system). While the

[53] `https://research.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html`

BVH format is primarily utilized for encoding point-cloud data, our extension considers the base rotation of joints to avoid twisting artifacts, in which the joint is oriented to the right position but twisted along its bone axis. In this format, the offset rotation rotates the joint so that its bone axis (y-axis) points along the bone to its main child. Thus, the format allows for a common understanding of joint positions and orientations (e.g., when the palm is facing upwards and when downwards). For the reference implementation, its basic structure (hierarchy tree) and base rotations are fixed: The joints of the exchange skeleton are orientated in a way that the x-axis is pointing in the direction of the main interaction (e.g., head x-axis pointing forward, wrist x-axis pointing in the direction of the palm). Hence, the skeleton is defined by all its joints and offset rotations up to the avatar's scale, which can change for different agents. Figure 5.3 shows a visualization of the exchange skeleton. The MMUs are responsible for retargeting the motion to the specific intermediate skeleton, considering the scale of the avatar in their synthesis as well. A retargeting service is provided and detailed in Section 5.2.

While the `MJoint` definition of exchange-skeleton joints is intuitively understandable and could be used to transmit the animation directly, its encoding is inefficient for motion transfer. During development, several options were evaluated. The recursive definition of the hierarchy by directly linking children in the `MJoint` struct is the most inefficient approach, requiring the most encoding and transmission time. Flattening the structure into a list is more efficient, but still not real-time capable. Similar to the BVH file format, the animation data can be extracted in depth-first order using a flat list of float values denoting the rotations and translations of individual joints. BVH relies on Euler angles or axis-aligned angles, which are easy to understand and allow a simple way to lock the rotation around specific axes, which would already enable realistic knee, elbow, and finger joints. However, Euler angles can suffer from gimbal lock and are not unique, as multiple Euler angles can describe the same rotation. Thus, the quaternion rotations are utilized directly. While this does inhibit the locking of individual axes directly in the animation representation, it also creates a more stable animation with little performance overhead. Adding an optional list defining a joint mask further optimizes the performance by only transmitting the animation data of the utilized joints. The full definition of the data structure is shown in Listing 5.5.

Adapters

Creating a single TCP (transmission control protocol) endpoint for each MMU is suboptimal in multiple ways. First, each MMU would occupy a single networking port. Second, synchronization of various components, mostly scene synchronization, would have to be performed for each MMU individually. Therefore, the concept of adapters is introduced to reduce computational overhead and ease the implementation of MMUs. An adapter is a language-specific component (e.g., C# Adapter) that loads MMUs, manages them, and redirects access accordingly. In addition, it synchronizes and buffers the scene. Most of the adapters load MMUs from the hard drive dynamically. This includes the C# Adapter, Unity Adapter, and C++ Adapter. Thus, the adapter does not depend on a specific MMU but can operate with any MMU for its specific language. The

Listing 5.4: Definition of the `AvatarPosture` and `MJoint` structures of the exchange skeleton.

```
struct MAvatarPosture
{
    1: required string AvatarID;
    2: required list<MJoint> Joints;
}

struct MJoint
{
    1: required string ID;
    2: required MJointType Type;
    3: required MVector3 Position; // offset in a joint local coordinate system
    4: required MQuaternion Rotation; // rotation in a joint local coordinate system
    5: optional list<MChannel> Channels; // animation channels (x,y,z,w) for rotations
                                         // and/or (x,y,z) position
    6: optional string Parent;
}
```

Listing 5.5: `MAvatarPostureValues` are utilized to transmit the animation data during runtime.

```
struct MAvatarPostureValues
{
    1: required string AvatarID;
    2: required list<double> PostureData; // animation data in depth-first order defined
                                          // by joint hierarchy and channel information
    3: optional list<MJointType> PartialJointList; // optional list of joints for which
                                                   // the posture data defines the
                                                   // new animation pose
}
```

co-simulator interface is similar to the MMUs, but offers additional functionalities, particularly via the co-simulator access interface. Hence, it is not loaded in the same environment as regular MMUs but from a dedicated adapter, which is adjusted to the co-simulator.

Python MMUs require a specialized Python environments that can have contradicting requirements for their utilized packages and Python versions. Hence, the MMU developer must integrate the MMU directly into the adapter. The implementation is simplified such that an adapter can be created in only a few lines of code. Listing 5.6 shows the example code required to initialize and start the adapter with an example MMU.

Services

A service-based structure offers accessible solutions to the developer, as many motion synthesis methods require similar functionality, such as IK, path planning, or motion blending. For example, the IK solver from the Blender editor is made accessible via a service implementation. Hence, any MMU implemented in Unity, C#, or Python can access the IK service via the (Thrift) service interface. Several service interfaces are predefined to allow an exchange of service implementation without the requirement to change the MMU implementation. Hence, the Blender-based IK service can be easily replaced by a different implementation, such as the FinalIK plugin in Unity, as both services implement the same interface.

Listing 5.6: Example for the initialization of the Python MMU `MyExampleMMU` with the `PythonAdapter`.

```
import MOSIM.PythonAdapter as PythonAdapter
from scripts.MMU_implementation import MyExampleMMU

if __name__ == "__main__":
    with open(".\\data\\description.json", "r") as f:
        description = "␣".join(f.readlines())
    PythonAdapter.start_adapter([(description, MyExampleMMU)])
```

Listing 5.7: Base-interface for the services.

```
service MMIServiceBase
{
    map<string,string> GetStatus(),
    MServiceDescription GetDescription(),
    MBoolResponse Setup(1: avatar.MAvatarDescription avatar,
        2: map<string,string> properties),
    map<string,string> Consume(1: map<string,string> properties),
    MBoolResponse Dispose(1: map<string,string> properties),
    MBoolResponse Restart(1: map<string,string> properties),
}
```

The `MMIServiceBase`-Interface (see Listing 5.7) defines a generic service with the base methods any service must implement. `GetStatus` provides meta information on the status (e.g., status, number of connected clients, update latency, etc.). `GetDescription` provides the service description, which is similar to the MMU description (see Listing 5.3) and provides meta-information on the service. The `Setup` method is utilized by the MMUs to initialize the service and provide the information on the simulated avatar and any other configuration required by the service. The `Consume` method is a generic interface that is utilized primarily during service development. A new service can be developed by implementing the base service interface and using the `Consume` interface for communication without defining a new specialized service interface. As `Consume` uses generic strings to pass information, all data must be serialized to a string before sending, which is computationally slow. Hence, it is recommended that a specialized service interface is defined in Thrift and implemented before publishing the service. In the following, several specialized service interfaces are presented.

Retargeting Service

The retargeting service is essential if the MMU does not perform custom retargeting to the MOSIM internal skeleton. The retargeting service utilizes the retargeting algorithm presented in Section 5.2. Listing 5.8 shows the definition of the retargeting service interface. It requires a reference posture in an upright T-pose, facing in a positive Z-direction with the arms and hands parallel to the ground. The legs should be straight without any angle. The retargeting service is not stateless but gets initialized with this reference posture in the `SetupRetargeting` method. Initial testing with MMU developers in the MOSIM research project showed that providing joint-local animation data in a serialized form such as the `MAvatarPostureValues` was too complicated. Hence, a more straightforward form using global joint information was utilized using a different interpretation

Listing 5.8: Retargeting service interface definition in Thrift.

```
service MRetargetingService extends MMIServiceBase
{
    MAvatarDescription SetupRetargeting(1:MAvatarPosture globalTarget),
    MAvatarPostureValues RetargetToIntermediate(1:MAvatarPosture globalTarget),
    MAvatarPosture RetargetToTarget(1:MAvatarPostureValues intermediatePostureValues),
}
```

Listing 5.9: Inverse kinematics service interface definition in Thrift.

```
service MInverseKinematicsService extends MMIServiceBase
{
    MIKServiceResult CalculateIKPosture (
        1: MAvatarPostureValues postureValues,  // input posture information
        2: list<MConstraint> constraints,  // spatial target constraints
                                           // (e.g. for the hands, feet, head)
        3: map<string,string> properties),
}
```

of the values within an `MAvatarPosture`. In this global representation, the data structure of an `MAvatarPosture` and `MJoint` (see Listing 5.4) is not interpreted in a joint local coordinate system, but the position and rotation are directly provided in the global coordinate system. This way, a higher usability was favored over a smaller transmitted data size.

After initialization, poses can be retargeted to and from the intermediate skeleton. To retarget to the intermediate skeleton, the current posture is provided as an `MAvatarPosture` in a global representation (see above). As a result, a single intermediate skeleton posture is provided as `MAvatarPostureValues` and can be exchanged with other components inside the framework.

In the other direction, the intermediate pose as `MAvatarPostureValues` is provided to the retargeting service to retarget from the intermediate skeleton to the prepared ego-skeleton using the `RetargetToTarget` method. A `MAvatarPosture` in global representation is returned in the representation of the ego-skeleton. The reference implementation of the retargeting service is detailed in Section 5.2.

Inverse Kinematics Service

Many applications require the use of an inverse kinematics solver to adjust a pose to reach a specific spatial target. The interface of the inverse kinematics service within the MOSIM Framework is shown in Listing 5.9. The solution is computed using the initial posture provided as `postureValues`. The end effector (e.g., left or right hand) and the target positions and rotations are defined in the `constraints`. If the target is an object, the object constraints (e.g., grasp constraints) are utilized. Before usage, the service is initialized with the standard `Setup` method. There are multiple implementations of the IK service: as Unity services, using the default and the FinalIK plugin, and as a Python service using the Blender IK solver. All implementations support full-body IK solving. While the FinalIK solver produces the best results, the Blender IK solver is an open-source component that is free to use.

Listing 5.10: Path planning service interface definition in Thrift.

```
service MPathPlanningService extends MMIServiceBase
{
    MPathConstraint ComputePath (
        1: MVector start,
        2: MVector goal,
        3: list<MSceneObject> sceneObjects,
        4: map<string,string> properties),
}
```

Path Planning Service

Path planning is another task that many motion synthesis approaches require. This includes 2D path planning for locomotion tasks, 2.5D path planning to incorporate different height levels (e.g., stairs), and 3D path planning for complex navigation (e.g., reaching around an object). Although these tasks can be directly implemented in the MMU utilizing the synchronized scene for planning, in many cases the separation of global path planning and motion synthesis is beneficial. For example, different MMUs like walking, running, or side-stepping can reuse the path planning service. Listing 5.10 defines the path planning service interface within MOSIM.

The `start` and `goal` positions are dimensionless, allowing for 2D, 2.5D, and 3D navigation input. A list of scene objects can include only specific objects for the path planning (e.g., the ground and obstacles, but not background geometry). Additional properties include the dimensionality of the path planning, weights for different surface materials, maximum acceleration and rotation velocities, and more. The reference implementation utilizes a path-planning service based on the navigation mesh agent in Unity.

Motion Blending Service

Although game engines such as Unity provide extensive blending frameworks (blending trees), not all implementations can rely on these systems. As many computational methods require the ability to blend between different poses, a specific service interface is implemented in MOSIM (see Listing 5.11). The `Blend` method requires a start and target posture and a global blending coefficient (weight). Optional blending masks containing joint-specific blending coefficients can be utilized. An additional `BlendMany` interface is implemented to blend multiple weights in parallel.

Access Services

To access the scene, skeleton, and co-simulator, several dedicated services are defined, which are not implemented in their own program but are incorporated within their respective main components. For the scene, there is additional support for mirroring the content in adapters and services.

Listing 5.12 shows an excerpt of the definition of the scene access service. Via different getter functions, specific information can be queried from the scene. The full scene description can be

Listing 5.11: Posture blending service interface definition in Thrift.

```
service MPostureBlendingService extends MMIServiceBase
{
    MAvatarPostureValues Blend
        (1: MAvatarPostureValues startPosture,
         2: MAvatarPostureValues targetPosture,
         3: double weight,
         4: map<MJointType, double> mask,
         5: map<string,string> properties),
    list<MAvatarPostureValues> BlendMany
        (1: MAvatarPostureValues startPosture,
         2: MAvatarPostureValues targetPosture,
         3: list<double> weights,
         4: map<MJointType, double> mask,
         5: map<string,string> properties)
}
```

Listing 5.12: Excerpt of the scene access service definition in Thrift.

```
service MSceneAccess extends MMIServiceBase
{
    list<MSceneObject> GetSceneObjects(),
    MSceneObject GetSceneObjectByID(1: string id),
    MSceneObject GetSceneObjectByName(1: string name),
    list<MSceneObject> GetSceneObjectsInRange(1: MVector3 position, 2: double range),
    list<MCollider> GetColliders(),
    list<MAvatar> GetAvatars(),
    list<MMesh> GetMeshes(),
    list<MTransform> GetTransforms(),
    double GetSimulationTime(),
    MSceneUpdate GetSceneChanges(),
    MSceneUpdate GetFullScene(),
    MNavigationMesh GetNavigationMesh(),
    ...
}
```

received via the `GetFullScene` method. Depending on the scene's size, this method requires more computation time. Thus, it should only be utilized for an initial setup of a mirrored structure in its own executable. The `GetSceneChanges` method is used afterwards to update the own representation by only transmitting the change of the scene since the last frame update.

With the skeleton access service shown in Listing 5.13, information about the skeleton can be queried. Besides receiving the avatar description, the current pose can be set via `SetChannelData`. Afterwards, the interface supports either getting joint position and rotation information in the (joint) local coordinate system or in global space. Similarly, the joint position and rotation can be directly set in global or local space. After manipulating the skeleton, however, the avatar posture values must be recomputed via `RecomputeCurrentPostureValues`. This step is not done automatically; it requires more computation time and slows down the process whenever multiple joints are repositioned via the interface.

Via the co-simulation access service, information on past instructions and poses can be queried, new instructions can be provided, and running instructions canceled. As this is essential for motion control, it will be further discussed in Section 5.3.2.

Listing 5.13: Excerpt of the skeleton access service definition in Thrift.

```
service MSkeletonAccess extends MMIServiceBase
{
    MAvatarDescription GetAvatarDescription(1: string avatarID),
    void SetChannelData(1: MAvatarPostureValues values),
    MAvatarPosture GetCurrentGlobalPosture(1: string avatarID),
    MAvatarPosture GetCurrentLocalPosture(1: string avatarID),
    MAvatarPostureValues GetCurrentPostureValues(1: string avatarID),
    MVector3 GetGlobalJointPosition(1: string avatarId, 2: MJointType joint),
    MQuaternion GetGlobalJointRotation(1: string avatarId, 2: MJointType joint),
    MVector3 GetLocalJointPosition(1: string avatarId, 2: MJointType joint),
    MQuaternion GetLocalJointRotation(1: string avatarId, 2: MJointType joint),
    void SetGlobalJointPosition(1: string avatarId, 2: MJointType joint,
        3: MVector3 position),
    MAvatarPostureValues RecomputeCurrentPostureValues(1: string avatarId),
    ...
}
```

Instructions and Constraints

Unlike standard animation systems, MMUs do not provide a single animation sequence but a full animation model. Consequently, developers do not need to provide individual blending weights but the higher-level information controlling the action like the target object, the walk target, or which screw to place in which hole. An instruction system was derived to enable developers to describe how the different actions should be performed and with which goal. The instruction interface and data types are kept generic, allowing for the specification of many actions. Listing 5.14 shows the definition of this data type.

Each MMU implements a single `MotionType`, and thus, by specifying the type rather than the specific MMU, the co-simulator can select the most fitting MMU for a given task. In addition, this allows for an easy replacement of an MMU without changing the agent behavior description. For example, a simple locomotion MMU can be utilized first for a general proof of context, and a more complex MMU, e.g., based on neural networks, can be added later. Motion generation can be further detailed by defining `Constraints` (e.g., goal position, goal direction, etc.) and `Properties` (e.g., velocity, style, etc.).

Similar to BML [Kop+06], start and end conditions are utilized by the co-simulator to schedule the execution of actions. This scheduling enables the combination of simpler MMUs to generate a more complex behavior. For example, by scheduling MMUs for retrieving an object, placing an object, grasping an object, and utilizing a screwdriver by rotating the wrist, the whole animation sequence using a manual screwdriver can be scheduled and simulated with more generic MMUs. An example of the syntax that is used to defined conditions is shown in Listing 5.15.

Several constraint types are defined. As Thrift lacks the possibility of inheritance, the struct `MConstraint` (see Listing 5.16) contains optional fields for the specific constraint types, allowing the gathering of different constraints in a single list in the instruction. Most important is the `MGeometryConstraint`, specifying geometric (position, rotation) constraints in cartesian space. By combining multiple geometry constraints, an `MPathConstraint` can be defined, and by

combining it with joint information, an `MJointConstraint`. When interacting with objects, the object itself contains object-specific constraint information, such as where to press the traffic light button or how to hold the screwdriver. A comprehensive annotation tool was developed to assist designers with specifying the constraints (see Figure 5.13).

Behavior Model

The MOSIM Framework's primary target is combining multiple simulation models into a coherent agent simulation. However, it is not an agent simulation framework, as it does not simulate the agent's intention, goals, or plans. A higher-level agent simulation and reasoning system provides the required control for the MMUs. Nevertheless, this reasoning system is not a geometric solver, as the geometric execution of the motion is the task of MMUs. For example, the behavior system might define which object needs to be retrieved and placed at which position (e.g., on the desk), but it does not prescribe the way, how the object must be grasped, and which path needs to be taken to the goal position.

The main interface to exert control over the simulation is the `MCoSimulationAccess` (Listing 5.17), a Thrift service implemented and started by the co-simulator. The actions are controlled via `MInstructions` (Listing 5.14), and the service provides interfaces to assign and abort instructions. The methods for `GetBoundaryConstraints` and `CheckPrerequisites` of the co-simulator are exposed to enable the behavior model to receive the required boundary constraints from the MMUs themselves and do not require any direct access to the MMUs or adapters.

The co-simulation access provides an event system for the behavior model to register a callback function. Whenever an MMU throws an event, for example to notify the start or end of an execution, the callback method receives the event. Additional methods to receive the history of events are also provided.

Intensive work was conducted to allow the combination of the MOSIM Framework with the AJAN framework for agent simulation [Ant+19] within the MOSIM and AIToC research projects. AJAN is a behavior-tree-based simulation framework that utilizes a dynamic knowledge base with a SPARQL (SPARQL protocol and RDF query language) interface to access an RDF (resource description format) triple database. Behavior tree nodes can access other tools via HTTP (hypertext transfer protocol) and MQTT (message queue (MQ) telemetry transport), for example, to gather the real-time state of real components. Action nodes utilize the `MCosimulationAccess` interface to send instructions to the MOSIM Framework.

In collaboration with the companies ZF Friedrichshafen AG and cogniBIT GmbH, a second proprietary behavior model (WalkBot) for simulating pedestrian agents in the CARLA driving simulator [Dos+17] was integrated. While AJAN establishes an indirect control of motion, the WalkBot establishes direct control, providing directional locomotion and gaze input for every frame.

Listing 5.14: Definition of the `MInstruction` datatype in Thrift.

```
struct MInstruction
{
    1: required string ID;
    2: required string Name;
    3: required string MotionType;
    4: required string AvatarID;
    5: optional map<string,string> Properties;
    6: optional list<MConstraint> Constraints;
    7: optional string StartCondition;
    8: optional string EndCondition;
    9: optional string Action;
    10: optional list<MInstruction> Instructions;
}
```

Listing 5.15: Example of the temporal scheduling of different MMUs to screw with a manual screwdriver.

```
graspInstruction.StartCondition    = retriveInstruction.ID + ":" + "AnimateFingers";
placeInstruction.StartCondition    = retriveInstruction.ID + ":" + "end";
utilizeScrewdriver.StartCondition = placeInstruction.ID   + ":" + "ObjectPlaced";
placeInstruction.EndCondition      = utilizeScrewdriver.ID + ":" + "end";
returnInstruction.StartCondition   = utilizeScrewdriver.ID + ":" + "end" + "+ 0.05";
graspInstruction.EndCondition      = returnInstruction.ID  + ":" + "ObjectPlaced";
```

Listing 5.16: Definition of the `MConstraint` datatype in Thrift.

```
struct MConstraint
{
    1: required string ID;
    2: optional MGeometryConstraint GeometryConstraint;
    3: optional MVelocityConstraint VelocityConstraint;
    4: optional MAccelerationConstraint AccelerationConstraint;
    5: optional MPathConstraint PathConstraint;
    6: optional MJointPathConstraint JointPathConstraint;
    7: optional MPostureConstraint PostureConstraint;
    8: optional MJointConstraint JointConstraint;
    9: optional map<string,string> Properties;
}
```

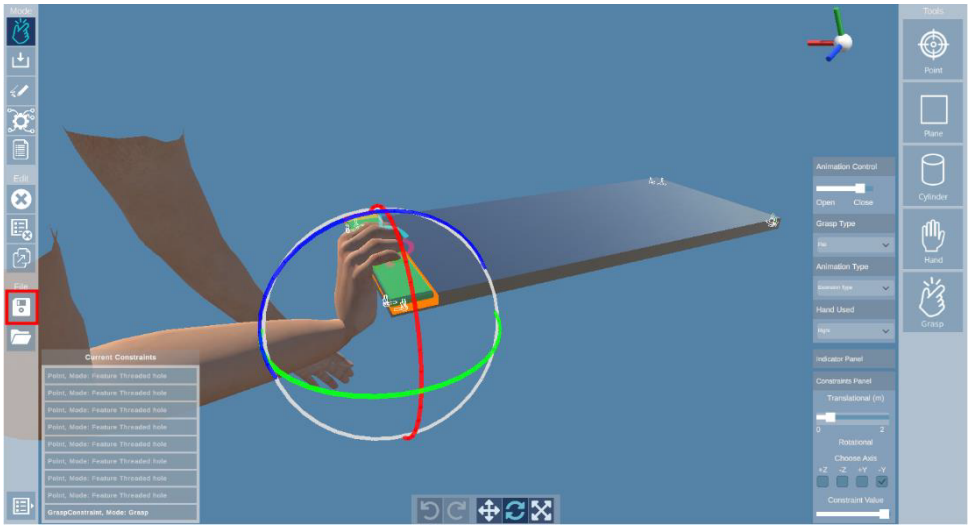


Figure 5.13: A constraint annotation tool was developed to simplify the annotation of object-specific constraints. Here, the grasping constraint of how to hold the shelf is annotated. First, the grasp type is selected (grasping a flat object), then the grasping target is positioned (green box), and the translational and rotational degrees of freedom to grasp this object are defined. The constraint is stored within the `MSceneObject`.

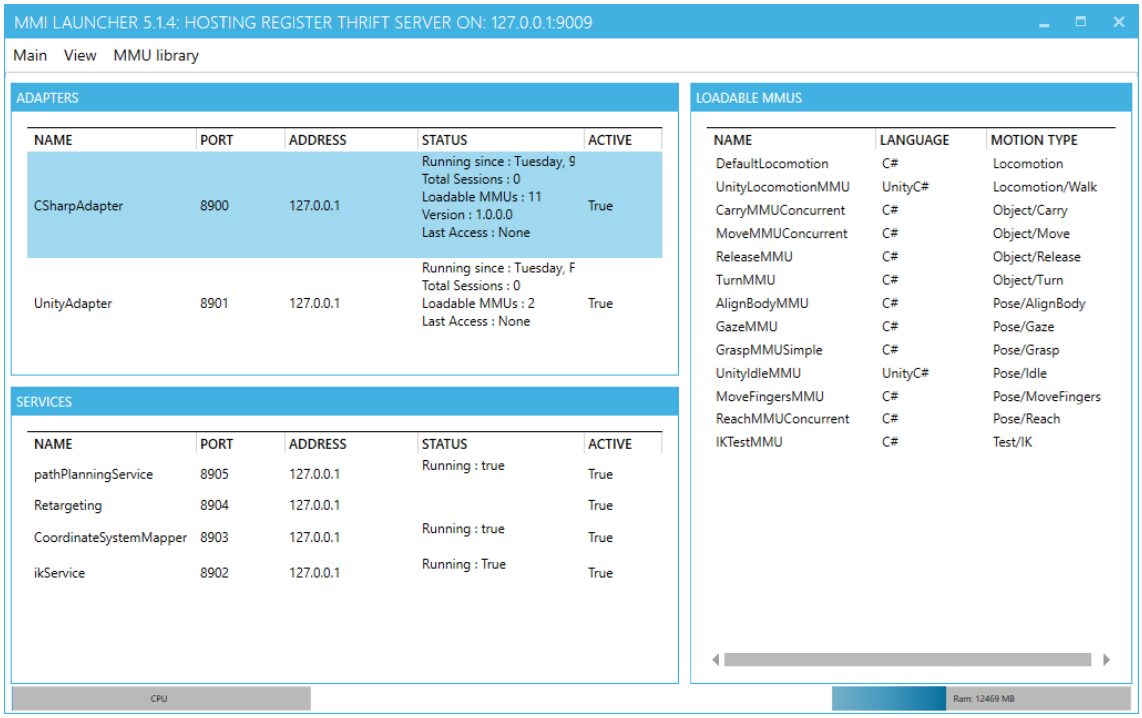


Figure 5.14: The MMI-Launcher with the available adapters (top left), hosting the MMUs for the simulation (right) and the available services (bottom left).

However, the MOSIM Framework itself does not always require a specific behavior model. The simplest form of a behavior model is a list of action sequences performed without any ability to react to dynamic changes in the environment. This is useful not only for debugging and visualization but also for generating static animation sequences.

MMI-Launcher

The MMI-Launcher is the central user interface to start and monitor the MOSIM Framework. It implements the `MMIRegisterService` (Listing 5.18), which can be queried to receive information on the available MMUs and Services. There are two versions of the MMI-Launcher. A server

Listing 5.17: Excerpt of the cosimulation access service definition in Thrift.

```
service MCoSimulationAccess extends MMIServiceBase
{
    MBoolResponse AssignInstruction(1: MInstruction instruction,
        2: map<string,string> properties),
    MBoolResponse AbortInstruction(1: string instructionID),
    list<MConstraint> GetBoundaryConstraints(1:MInstruction instruction),
    MBoolResponse CheckPrerequisites(1:MInstruction instruction),
    MBoolResponse RegisterAtEvent(1: MIPAddress clientAddress, 2: string eventType,
        3: string avatarID);
    MBoolResponse UnregisterAtEvent(1: MIPAddress clientAddress, 2: string eventType,
        3: string avatarID);
    list<MCoSimulationEvents> GetHistory(1: string eventType, 2: string avatarID),
    MCoSimulationEvents GetCurrentEvents(1: string avatarID);
}
```

executable optimized to run without a graphical interface and a Windows implementation with a graphical interface (see Figure 5.14). The launcher hosts the registry service outlined in Listing 5.18.

The graphical launcher allows the automatic starting of all components (adapters, MMUs, and services), making it the primary tool for hosting the framework on a local Windows machine. It shows the status of all loaded components and provides developers with detailed information on the available MMUs. In addition, the framework can be configured in different network environments by configuring the port utilization in the launcher settings.

Listing 5.18: Thrift interface of the MMI register service.

```
service MMIRegisterService
{
    list<MAdapterDescription> GetRegisteredAdapters(1: string sessionID),
    list<MServiceDescription> GetRegisteredServices(1: string sessionID),
    map<MMUDescription, list<MIPAddress>> GetAvailableMMUs (1: string sessionID),
    MBoolResponse RegisterAdapter(1: MAdapterDescription adapterDescription),
    MBoolResponse RegisterService(1: core.MServiceDescription serviceDescription),
    ...
}
```

5.3.3 Engine Integrations

The target engine is not part of the MOSIM Framework itself but the component in which the simulation of the MOSIM Framework is utilized and visualized. In the course of our projects, we have integrated the MOSIM Framework primarily into two engines: Unity and Unreal Engine. For the integration, three major aspects need to be considered: (i) central tick integration, (ii) motion transfer, and (iii) object synchronization.

Central Tick Integration. The first step is integrating the MOSIM Framework into the central simulation update (tick) of the engine. Game engines and simulators progress time in discrete steps, and at every step, a corresponding update method is called. The MOSIM Framework must be integrated into this update loop. The MOSIM scene must be synchronized with the adapters and co-simulation for each update. The `DoStep` method of the co-simulation must be called to receive the simulation of the virtual agent, and the resulting pose and scene updates need to be applied. Game engines usually provide the time that is simulated as the actual real-world elapsed time since the last rendering update; however, this is not necessarily the case for the simulation of the MOSIM Framework. Three different times in a system can be identified.

1. The 'observation time' that elapses for the perceiving human user watching the display since the last update.
2. The 'render time' required to update the simulation and generate the next image.
3. The 'simulated time' that the simulation was generated for.

'Real-time visualization' means that all three times are equal, and the user perceives the animation for the same amount of time it was simulated. Whenever rendering takes longer than the observation time, the simulation lags behind, and the user perceives the animation slower than intended. In particular, this can occur when MMUs require longer simulation times and should be avoided. If it cannot be avoided, the individually rendered images need to be combined into a video in a post-processing step if smooth visualizations are required. In several use cases, changing the simulated time is relevant. By increasing the simulated time, the whole simulation can be sped up to simulate and visualize more activities in a shorter amount of real-world time. By decreasing the simulated time, the simulation can be slowed down and provide a more focused view of single motions for an observer. Most MOSIM demonstrations are executed using the 'real-time visualization' settings.

Motion Transfer. For every engine, the ability for mesh transformations based on kinematic joint controllers is a minimal requirement. Skeleton-based mesh deformations are well-studied with a wide variety of models. Lewis et al. [LCF00] propose one algorithm and describe other approaches comprehensively. However, the specific implementation of skeletons differs in engines. Besides differences in the coordinate system handedness, the joints are not necessarily rotated in the same direction in their zero poses through post-processing during the model import. Additionally, engines tend to optimize and adjust the skeletons. Thus, poses cannot be copied from Unity and pasted on the same character in Unreal Engine and Blender. There is excellent support for the static integration of animations via the file system (e.g., using the FBX file format), but there is almost no support for runtime animation transfer. The retargeting service can be used to retarget the motion from the intermediate to the target skeleton of the animated character. In most engines, it is possible to change the rotations and locations of joints procedurally with custom implementations.

Object Synchronization. Animations in free space are great for exploring new animation methods but are not useful in practical applications. Virtual agents must perceive the objects in their environment and control them to achieve the targeted behavior. For this purpose, bi-directional synchronization between the objects in the environment and their representation in the MOSIM Framework is required. By implementing the scene access services (see above, Listing 5.12) and the associated data types (e.g., `MSceneObject`, `MMesh`, etc.), this synchronization is achieved.

Coordinate System Alignment. It is essential to handle different coordinate systems correctly. Besides considering the handedness of the coordinate system for rotations, an upward movement should remain upwards, and a left movement should continue towards the left. Thus, it is not simply possible to match the y-translation value of one engine to another, as it might mean "up" in one engine (Unity), "right" in another (Unreal Engine), and "forward" in a third (Blender). By overlaying the axes, the mapping can be identified. Figure 5.15 shows the example of three different coordinate systems aligned with their semantic meaning. Thus, it is trivial to identify the mapping of a MOSIM vector $v = x, y, z$ to an Unreal vector $v' = z, x, y$. The same mapping can be applied for rotations with a MOSIM quaternion rotation $q = x, y, z, w$ expressed as an Unreal quaternion rotation as $q' = z, x, y, w$. If the handedness of the systems does not match, as with

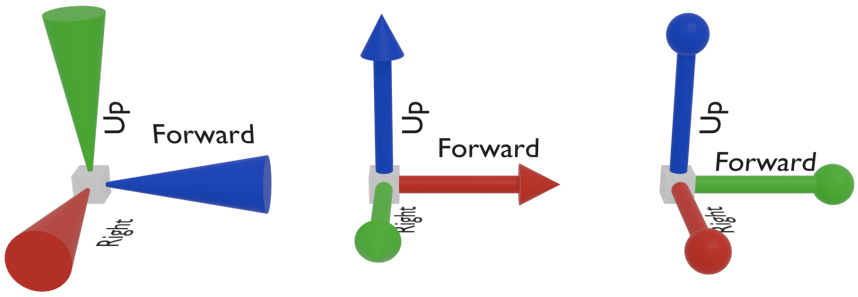


Figure 5.15: Different coordinate systems (from left: Unity/MOSIM, Unreal Engine, Blender. Red denotes the x-axis, green the y-axis, and blue the z-axis of each coordinate system. The MOSIM and Unreal Engine coordinate systems are left-handed, while the Blender system is right-handed.

the Blender coordinate system, the w-component needs to be negated, and the quaternion needs to be normalized. This step is not required in the example because MOSIM and Unreal Engine coordinate systems are both left-handed.

5.3.4 Separation of Responsibility

When the MOSIM Framework was developed initially, it was intended to simulate manufacturing processes in automobile production. As such, the MTM-1 (Methods-Time Measurement) [MSS48] building blocks were considered. To follow the MTM paradigm, tasks were separated into fine-grained actions, such as reaching, grasping, or walking. By combining the MTM actions, the simulation of a wide variety of more complex tasks was targeted. However, in practice, this approach resulted in overly complicated behavior trees even for very simple tasks, and the resulting animation was not natural, as the individual MMUs were not aware of the semantics of the overall animation. Thus, a new set of MMUs were derived, which are less motion-focused and more task-focused. These include actions like retrieving an object, placing an object, screwing a screw by hand, screwing a screw with a screwdriver, etc. While this naturally creates redundancy in the animation models, it simplifies action sequences and their utilization by a behavior model like AJAN [Ant+19], making its use more transparent.

The more complex MMUs, however, also require more complex constraints. All the necessary constraints can be gathered by calling the `GetBoundaryConstraint` methods for each MMU, ensuring that the behavior only relies on symbolic information and does not need to perform intensive geometric reasoning and transformations. Object-specific constraints can be received from the `MSceneObjects`.

We take as an example the case of screwing a screw with a manual screwdriver.

1. The **screw MMU** requires the screw-head orientation and position, the screw length, and the tooltip of the screwdriver to animate the screwing motion using a manual screwdriver. This information is object-dependent and is stored within the object constraints. Thus, only the ID of the screw and screwdriver objects need to be provided in the instruction. For this example,

we assume the screw was already placed at the correct location (e.g., by hand screwing). However, before the screw MMU can be utilized, the screwdriver first needs to be retrieved, grasped, and placed in the correct position.

2. The **grasp MMU** requires information on the grasping pose around the screwdriver's handle, which is stored within the object constraints of the screwdriver.

3. The **retrieve MMU** requires the goal position of the wrist, which it can receive using the `GetBoundaryConstraints` method of the grasp MMU.

4. The **place MMU** requires the goal position of the wrist, which places the screwdriver at the correct position. This goal position can be computed by combining information from the `GetBoundaryConstraints` methods of the screw and grasp MMU.

5. If the character is in a completely different location, the **walk-to MMU** requires the information of where to send best to retrieve the screwdriver, which it can request from the `GetBoundaryConstraints` method of the retrieve MMU.

The flow of information in such a task sequence is one-directional (Screw - Grasp - Place - Retrieve - WalkTo). While it may seem overly complicated, requesting the necessary constraints is possible in a purely symbolic state. Thus, the behavior model does not need to compute geometric constraints but only requires knowledge about the symbolic relation between different actions to concatenate multiple actions into a complex sequence. This is particularly useful for behavior models like AJAN [Ant+19], which operates in this symbolic space and utilizes an advanced knowledge database. Figure 5.16 displays representative figures from such a complex sequence simulating the partial assembly of a shelf, the final demonstrator for the ITEA4 Project AIToC.

Even though the MMUs are more object-specific, they are still modular and combinable. While it is possible to further aggregate actions to a more complex MMU, it is not always recommended. For example, the naïve plan for the construction of the shelf requires first picking up the screw, then screwing it by hand, and fastening it with a manual screwdriver. As a result, the screwdriver is picked up and put down multiple times, resulting in a very inefficient workflow. By reordering the actions, a more fluent workflow can be derived without needing to adjust the MMUs. With hand-crafted super-MMUs simulating the whole process, this type of behavioral optimization would be impossible.

5.3.5 Update Instruction

While the primary goal of instructions is to control the motion generated in MMUs, they are first processed by the co-simulator to ensure that preconditions are met and to schedule the exact timing of the start of the MMU execution. The constraint solving and scheduling of the co-simulator is primarily optimized for an indirect control of motion, meaning the number of instruction assignments is significantly smaller than the number of frame updates. However, some behavior models require direct motion control by updating the control input every frame. In this case, the

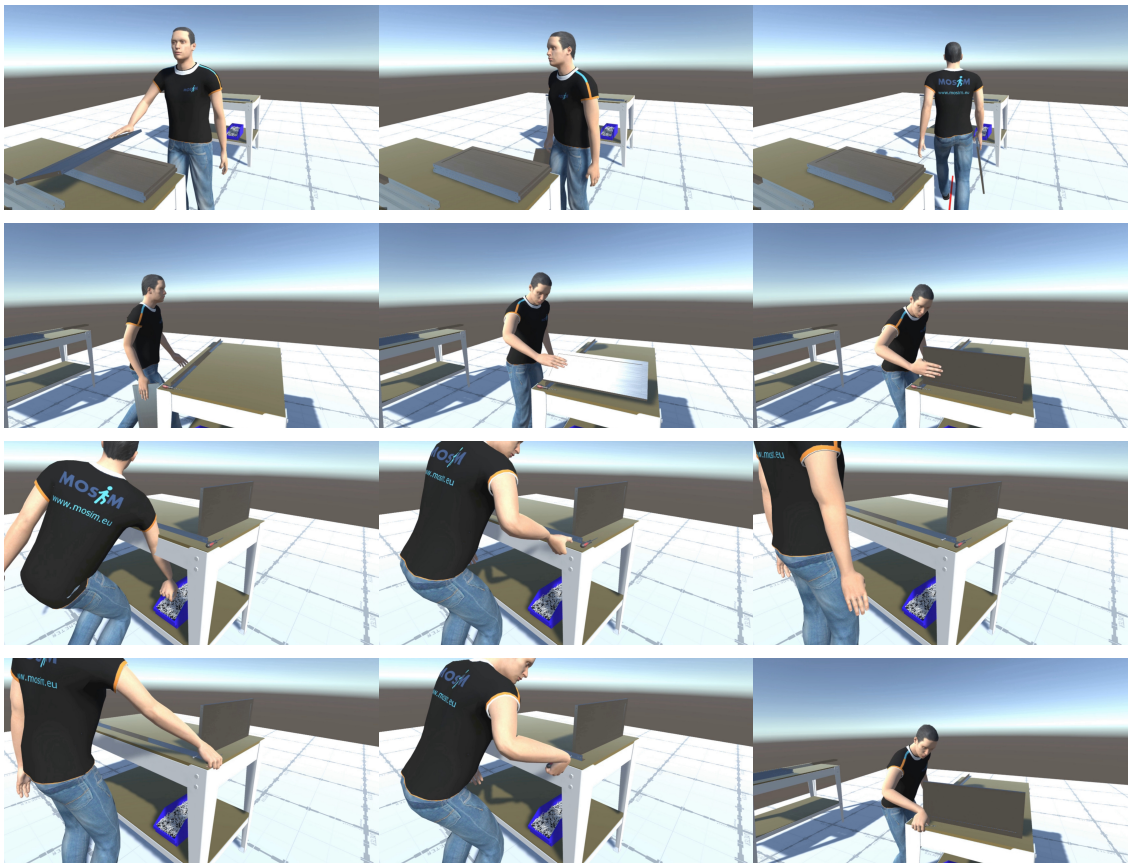


Figure 5.16: Sequence images from a simulation using the MOSIM Framework to build a virtual shelf. As examples from the actual manufacturing domain are difficult to publish due to intellectual property considerations, a popular shelf from a Swedish furniture supplier was utilized for demonstration.

overhead of the co-simulator is slowing down the execution of motion significantly. Therefore, the concept of update instructions is introduced to enable continuous motion control before every frame generation. A second instruction with the same ID is sent to the co-simulator to update the initial instruction. The co-simulator then circumvents any constraint checking for the update instruction and directly forwards it to the MMU. Using the update instructions, the framework can be controlled using a Python interface, simulating the motion using a neural network within Unity and visualizing it within the Unreal Engine at 30-60 fps on a standard gaming PC. In comparison, the animation does not run faster than 20 fps with regular instructions.

5.3.6 Evaluation

Runtime

The most frequently used functions are `AssignInstruction` and `DoStep`. Their runtime performance was evaluated with a single agent in an empty Unity scene. Using a reflector MMU implemented in C#, which does not do any computation but returns a constant pose, the base latency of both commands can be measured. As the co-simulator is integrated into the Unity target

engine and different pre-checks are implemented within the update loop, the runtime difference between game-engine ticks without MOSIM commands and with MOSIM commands are contrasted. The evaluation is executed on a gaming PC (AMD Ryzen 9 16 Core CPU, NVIDIA RTX 3080 GPU, 64 GB RAM), and all MOSIM components communicate via a loopback connection (localhost). Both commands are run 1000 times and are compared against the average frame time of 1000 frame updates without any command. Game tick rates are compared as the function trigger code in several game components, and the scene graph is changed. Compared to a standard game tick update, assigning an instruction requires an additional 1.361 ms per frame and 1.242 ms in case of an update instruction. Invoking the `DoStep` method of the reflector MMU requires an additional 0.466 ms per frame.

A more realistic example can be observed when running the Unity-Idle MMU, which uses a blend tree to play an idle animation. This test utilized the standard Unity adapter as a deployed executable. Calling this MMU via the `DoStep` method requires 13.913 ms on average (71.88 fps) to generate a new frame. This includes 1.631 ms for the transfer of poses on average. Using the update instructions, a single instruction assignment required an additional 4.48 ms. The behavior when assigning instructions differs depending on the adapter implementation and MMU. Unlike the debug adapter, the default Unity adapter creates a new session with a new instruction ID every time a new instruction is received to allow parallel execution of the same MMU (e.g., to reach with the left and right hand in parallel). As a result, executing multiple instructions after each other increases computation time significantly, gradually increasing the instruction assignment time with each new instruction, resulting in over 80 ms of additional computation time per instruction after only 100 assigned instructions and an average of 43.54 ms per instruction.

Memory Consumption

The MOSIM Framework's memory consumption was measured using the deployed framework. While it depends on the actual implementation of the MMUs, the version utilized within the research projects using different Unity-MMUs and services require a total of 475 MB of memory. While components utilizing larger frameworks (e.g., Unity) and containing data (e.g., animations) require more memory, simple .NET applications (e.g., retargeting and co-simulation) require only a minimal amount of memory. While the Unity Adapter with several animation-based MMUs required the most memory (102 MB), more specialized Unity services like the Unity IK Service, a dedicated Unity MMU, and the Unity Path Finder required about 40-70 MB each. The Windows MMILauncher and C# Adapter require about 50-60 MB and highly specialized services like the Co-Simulation Standalone and Retargeting Service between 4-10 MB. Several other processes, like the crash handlers, minor services, and logging, required an additional 73 MB in total.

5.3.7 Discussion

The MOSIM Framework is a lightweight, well-integrated simulation framework supporting the integration of various motion simulation techniques. While it is not designed to be used in high-performance games or shipped as a part of a game, it is sufficiently performant and usable for professional simulation applications. It allows the recycling of motion simulation models and animation tools. It was successfully used in simulations of factory workers in the final assembly of a digital factory and for the simulation of pedestrian agents in pre-crash simulations. The MOSIM Framework was successfully integrated into Unity, Unreal Engine, and Blender, with additional support for native C#, C++, and Python applications. However, there are still possibilities to increase the framework's performance, stability, and usability.

At the time the MOSIM Framework was conceived, detailed control over the generated motion was requested. This had several implications on the retargeting and co-simulation requirements. Primarily, the co-simulator should only perform minor adaptations of the generated motion and not contain its own assumption on the human body model. From its current use, it would be beneficial if the co-simulator would have its own body model, adapting incoming animation to provide a clean and plausible output. This body model should not only provide a reference skeleton but also constrain the range of motion and have its own dynamics. As such, an MMU would propose poses to the body model that would naturally follow these poses. By adding standard human functions like a breathing cycle, eye blinking, and a natural variation in idle movements (e.g., weight shifts, twitching of fingers, head-scratching, etc.), the overall realism would be increased without increasing the complexity of MMUs. Both, a physically based and a kinematic digital human model could improve the quality of motion and avoid unnatural poses and transitions.

The framework is still a prototype and not yet a product. Hence, some issues regarding performance and memory management remain. While these issues can be annoying and detrimental to an actual product, they do not inhibit the general functionality of the framework and, thus, are minor in their priority.

5.4 Chapter Discussion and Future Work

In this chapter, the MOSIM Framework and a real-time retargeting solution to transfer motion to different characters with different hierarchies and scales are presented. The systems shown here have been successfully utilized in various research projects, and since 2021, I have been the principal curator of the MOSIM Framework. In the research projects MOSIM (ITEA3, EU, 2018-2021), AIToC (ITEA 4, EU, 2021-2024), and TwinMap (BMWK, 2024-2026), they were utilized for the simulation of digital twins of factory workers. In the research project MOMENTUM (2022-2025) and our collaborations with the company ZF (2021, 2022), they were utilized to integrate a pedestrian simulation using a mixture-of-expert based generative motion model implemented within the Unity game engine to the driving simulator CARLA, implemented in the Unreal Engine. Here, the pedestrians can be controlled responsively by human user input (see Figure 4.12 for an example), by an indirectly controlling behavior model (e.g., AJAN [Ant+19]), or by a directly controlling behavior model (e.g., CogniBit's WalkBot utilized in a feasibility study funded by ZIM/BMWK, 2023).

Our approach enables the integration of pedestrian animation models into existing ADAS environments, answering the third research question: How can pedestrian movements inside ADAS environments be visualized? Given a working MOSIM endpoint, the generative models for pedestrian simulation can be integrated via the MOSIM Framework in the target environments (e.g., CARLA) that generate the sensor input, without requiring additional adjustments in the motion model.

5.4.1 Discussion

The MOSIM Framework is a practical tool to integrate different animation systems with an external behavior model in various applications. However, it is not optimized for deployment on a single-user machine as a commercial software application. Particularly in professional environments of larger companies, in which the IT departments have strict regulations on port availability and access rights, local hosting of services relying on network communication is sub-optimal, and a single-engine solution is more practical. However, almost all central components have been adjusted to run in a docker container on a remote server, allowing users to have on-demand access. Performance issues due to the reliance on an outdated version of Apache Thrift, a missing internal body movement, and, most of all, a lack of active users and MMU providers limit the usability of the framework.

Unity and Unreal Engine are the two biggest game engines, and their functionality is constantly improving. With the Unity Muse tool, text-based animation generation is accessible. With Microsoft's ONNX runtime library [dev21], the native integration of neural networks trained with Python-based libraries in Unity is feasible. Unity's ML-Agent tool provides ample capability for reinforcement learning-based animation controllers. Conversely, Unreal Engine 5 supports native

neural network import and allows the extension of its animation graphs with generative animation notes. The Unreal Engine Metahuman library has extensive facial capturing and animation features. Both engines support plugins for motion matching [Cla16], an extensible data-driven animation approach. However, issues remain with integrating external animations, online retargeting, and sharing data and scene information in real-time and across platforms.

5.4.2 Future Work

Motion models are the most important factor in creating convincing simulations. When simulating pedestrians, a limited amount of different actions are required, with most of them being related to locomotion (like walking, running, or waiting), social interaction (like gestures, waving, pointing), and simple interactions (like opening the door, standing up and sitting down, or pressing a button). However, other use cases are more complicated. For example, we identified over 100 different operations with additional sub-actions in the manufacturing use case. Manually defining all animations is tedious and inefficient. However, combined with more digital twins of other components (machines, environment, tools, etc.) and formalized task descriptions, an automatic MMU generation using reinforcement learning and imitation learning appears feasible and should be investigated in the future.

Besides automated MMU generation, more support for MMU scaling is required to enable MMUs to generate motion for arbitrary-sized character models. While parametric models like SMPL [Lop+15] have already built extensive groundwork, they restrict the number of possible skeleton hierarchies and character models and, hence, are suboptimal. Further extension of the database- and learning-based approaches – as outlined in Section 5.2.5 – and integrating it as a tool into the MMU generation pipeline would provide more usability to developers.

The MOSIM Framework will only achieve commercial viability if it runs as an on-demand service on a server. One of the most significant technical drawbacks in this regard is its reliance on Apache Thrift as a middleware. Since the release of Thrift version 14 in February 2021, there has been a switch from the .Net Framework to the newer .Net Core Framework, which Unity does not yet support. Hence, upgrading the middleware to a newer version will exclude Unity as a supported platform and thus, is not feasible. An in-depth investigation showed the feasibility of replacing the middleware with either a custom implementation or a gRPC (gRPC Remote Procedure Calls) and protobuf (protocol buffers) solution. Future work is required to implement this new communication layer.

In addition to the technical issues, no privacy model has been implemented, yet. All communication is sent via an unencrypted connection without session authentication, and session IDs are shared in plain text. While MMUs and services can be deployed as black-box applications, preserving their IP and skeleton hierarchy, there is a strong reliance on the target engine sharing information, geometry data, and state data to allow a smooth animation. As such, scene content, like the geometry information of objects, might not be sufficiently preserved.

Chapter 6
Conclusion and Future Work

This dissertation focuses on capturing and simulating pedestrian movements for virtual pre-crash simulations of autonomous vehicles. Three research questions are investigated: (i) how to capture realistic pedestrian movements, (ii) how to simulate the body movements of pedestrian agents, and (iii) how to visualize the movements in ADAS simulation environments.

In Chapter 3, the realistic capturing of pedestrian movements before and during street crossings is investigated. Multiple studies were conducted in real and in virtual reality environments using IMU-based motion capturing. While capturing motions in real environments results in more valid animations, the organizational overhead and post-processing effort make the paradigm non-optimal. Capturing movements in large-scale virtual reality environments allows for the safe conduction of highly structured experiments and automated post-processing. Cross-cultural studies with equal environmental conditions are feasible using the VR paradigm, and experimentation with high-risk groups like school children is viable. The captured motion can be utilized for generative motion modeling, providing additional environment information like the position on the street or the crossing intention. In addition to motion capture data, the experiments can be used to capture and extract additional information for path generation, saliency prediction, pedestrian behavior modeling, and pedestrian behavior analysis, thus further increasing the value of this paradigm.

In Chapter 4, different data-driven simulation methods for generative pedestrian motion synthesis are presented. These methods focus on maintaining and reproducing the inter- and intra-personal variance required to produce natural animations, especially in groups of people. Additional approaches for multi-action simulation are investigated, and a novel approach for natural avoidance simulations in crowds is presented. All approaches utilize motion capture data and thus can be trained on the data captured in the pedestrian experiments. However, the internal motion representation and the simulation environment do not necessarily match ADAS simulations and thus require additional integration effort.

In Chapter 5, the MOSIM Framework is presented, an open and modular framework for efficient and interactive simulation and analysis of realistic human motions in professional applications. It enables the separation of simulation and visualization, simplifying the integration effort for individual animation modules and enabling reusability of the same model for different target

applications. In addition, the chapter focuses on the motion transfer and retargeting functionality of the MOSIM Framework, as it is the central component for transferring motion between the generative motion model and the target character used for visualization. In combination with the generative animation models outlined in Chapter 4 and based on the motions captured in Chapter 3, realistic animations of virtual pedestrians in pre-crash situations can be integrated into driving simulators like CARLA.

Thus, in this dissertation, the whole path from capturing pedestrian motion over the training of generative motion models to its integration in driving simulators is investigated. The results of this dissertation can already be used to simulate pedestrian agents in pre-crash scenarios. The outlined approaches can be used additionally as a blueprint for new studies to gain an even better understanding of pedestrians and thus, better pedestrian agents.

6.1 Limitations

The data of the virtual reality studies are very detailed and can be automatically processed. While some of the results correspond to observations found in real environments by others, a direct correspondence to real crossing behavior was not evaluated. Participants can be biased by the experimental situation, behaving more rule-abidingly and safely than in non-experimental situations. The VR environment and the repetitive nature of the experiments could result in adapted behavior toward the environment and not reflect the real behavior in naturalistic environments. A direct comparison of pedestrian behavior in a real and a digital twin of a virtual street could highlight more details on the effect of the VR environment. However, experimental studies are intensively utilized in traffic psychology, and even if the exact decision criteria, such as gap acceptance, are not the same, it does not inhibit the usability for modeling and simulation. Therefore, more studies are necessary before the data and corresponding models can be used in productive systems to establish the validity of the recorded data.

This dissertation targets the simulation of pedestrian movements in pre-crash simulations. These simulations are only usable for the training and evaluation of ADAS systems if these systems can observe and utilize the cues provided by the body. A system, for example, that purely relies on the past position and orientation of the body will never benefit from a more realistic motion simulation. However, different cues in the body movements, e.g., upper and lower body rotation, gaze, or gestures, are essential for differentiating a crossing from a non-crossing intention [Spr+19a]. The generated agents have not yet been utilized to train and evaluate ADAS. Further research is required to establish the usability of more accurate pedestrian motion simulation models in the training and evaluation of ADAS.

6.2 Capturing of Critical Scenarios

The motions captured with participants in our experiments so far have focused on regular crossing scenarios of uncritical nature and primarily with regular, healthy adults. As the paradigms utilized in our experiments were novel, there was not enough data with a healthy participant group to motivate research in more critical and risk-related groups. Our results, however, lay the basis for such studies. Especially in virtual reality environments, there is no inherent risk to the participants other than the regular risk of using VR equipment. Our feasibility study (see Section 3.4) has shown that the paradigm can be extended for experiments with school children. While elderly people are generally less inclined to adopt new technology (e.g., [MSC22]), there have already been studies with the elderly using VR (e.g., [Gar+15]). Single individuals beyond 80 years have tried out our virtual reality setup successfully, and thus, experiments with the elderly in large-scale virtual reality environments seem possible. The only limitation with the current setup is with disabled participants. As the motion-capturing equipment requires specific calibration postures, participants must be able to perform these (e.g., standing still in an upright position, walking around for a few seconds, etc.). There are many conditions in which these postures cannot be achieved. Even so, with an adjusted motion-capturing system, a similar paradigm would most likely be feasible for safe crossing experiments with disabled participants.

Besides opening up experiments with a wider group of participants, virtual reality allows for experiments in close-collision and collision situations. In our VR experiment presented in Section 3.3, participants were asked to collide with an oncoming vehicle intentionally. While almost all participants were excited about the experience, getting hit by a virtual car caused visible effects of minor discomfort, like flinching or even jumping to the side. In the experiments, participants generally tried avoiding collisions. However, virtual reality allows the creation of situations of near misses and collisions intentionally by mimicking the perception errors of participants. For example, if a participant looks left, a vehicle can be spawned to the right that the participant has not yet seen. This mimics the effect of the participant overlooking the vehicle. Urgency can be modeled (e.g., by providing rewards) to the point where participants accept a high risk. Occlusion can be easily achieved, and the approach behavior in occluded environments can be observed. Distractions, like letting a child chase a butterfly, can be included without harming the participants. Future experiments will allow further investigation of pedestrian behavior in these situations and provide the data necessary to generate realistic simulations of these critical situations.

6.3 Controlled Simulation of Pre-Crash Situations

While macroscopic city simulations require naturally behaving pedestrian agents to simulate the environment accurately, more control is necessary in pre-crash simulations. Corner cases and rare events need to be created in a reproducible way. A perfect pedestrian agent might never collide with a vehicle after all. The captured data can help partially by providing trajectory examples for

different safety margins and thus with different criticality. However, the natural trajectories for the controlled simulation need to be optimized to ensure a collision with an approaching vehicle if the vehicle does not change its velocity and steering angle. The testing scenarios like NCAP [Eur17] define specific collision points with respect to the car front (e.g., central, 25% to the left, etc.). For the correct visualization, the pedestrian must follow the trajectory and be at the specific spatio-temporal location for the collision. Using procedural controllers with mixture-of-expert models as described in Section 4.2 and Section 4.3, as well as reinforcement controllers with latent motion models as described in Section 4.5 cannot ensure these spatio-temporal constraints. The primary reason is the limited time horizon of future states accessible to the model. Even though the reinforcement learning controller is trained to optimize the future reward, it does not guarantee to create an action sequence to reach the constraint.

New animation techniques might improve the accuracy. Especially diffusion models are a good candidate for future investigations. Although there are still restrictions due to their inference time, they can be trained to optimize the complete trajectory to ensure that spatio-temporal constraints can be fulfilled. Using guided diffusion, the trajectory can be formulated as a condition to guide the diffusion process to create an animation for a specific trajectory [Rem+23].

6.4 Size-dependent Motion Simulation

The data captured in our experiments described in Chapter 3 contains a multitude of different-sized participants. Although primarily containing locomotion, the data can be used to train a model that takes the character's size as an input. Size-dependent animation creation is feasible, especially when using motion VAEs with a conditional variational autoencoder as the "motor" of the animation [Rem+21].

To further increase variability, especially in cases where only a few actors are available, synthetic data augmentation can help to create sufficient training data for a proof of concept. In multi-action control, for example, for climbing animations or the animation of factory worker actions, this is especially important, as capturing and annotation create significant effort. Especially for these application areas, the proper simulation of actions is more important, as trivial corrections for size are not always applicable. It can be assumed that the style is natural enough if the augmentation only creates minor variations around the original data. As a result, the motion model could intrinsically generate the correct actions for different characters.

Bibliography

[Aba+15] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/ (Accessed: 7 Dec. 2024).

[Abe+20] Kfir Aberman et al. "Skeleton-aware networks for deep motion retargeting". In: *ACM Trans. Graph.* 39.4 (Aug. 2020), 62:1–62:14. DOI: 10.1145/3386569.3392462.

[AF02] Okan Arikan and D. A. Forsyth. "Interactive motion generation from examples". In: *ACM Trans. Graph.* 21.3 (July 2002), pp. 483–490. DOI: 10.1145/566654.566606.

[Agu+08] Edilson de Aguiar et al. "Performance capture from sparse multi-view video". In: *ACM SIGGRAPH 2008 Papers*. SIGGRAPH '08. Los Angeles, California: Association for Computing Machinery, 2008. DOI: 10.1145/1399504.1360697.

[AH20] Essam S. Almutleb and Shirin E. Hassan. "The Effect of Simulated Central Field Loss on Street-crossing Decision-Making in Young Adult Pedestrians". In: *Optometry and Vision Science* 97.4 (2020). ISSN: 1538-9235. URL: https://journals.lww.com/optvissci/fulltext/2020/04000/the_effect_of_simulated_central_field_loss_on.2.aspx.

[Ahn+18] Hyemin Ahn et al. "Text2Action: Generative Adversarial Synthesis from Language to Action". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 5915–5920. DOI: 10.1109/ICRA.2018.8460608.

[Akm+25] Ulan Akmatbekov et al. "Simulating Body Movements for Multiple Agent Avoidance". In: *2025 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 2025.

[Akm24] Ulan Akmatbekov. "Modeling Multiple Avatar Avoidance Movements". Master thesis. Saarland University, 2024.

[AL11] Andreas Aristidou and Joan Lasenby. "FABRIK: A fast, iterative solver for the Inverse Kinematics problem". In: *Graphical Models* 73.5 (2011), pp. 243–260. DOI: 10.1016/j.gmod.2011.05.003.

[Ale+23] Simon Alexanderson et al. "Listen, Denoise, Action! Audio-Driven Motion Synthesis with Diffusion Models". In: *ACM Trans. Graph.* 42.4 (July 2023), 44:1–44:20. DOI: 10.1145/3592458.

[AM19] Chaitanya Ahuja and Louis-Philippe Morency. "Language2Pose: Natural Language Grounded Pose Forecasting". In: *2019 International Conference on 3D Vision (3DV)*. 2019, pp. 719–728. DOI: 10.1109/3DV.2019.00084.

[Ame19] American Automobile Association (AAA). *Advanced Driver Assistance Technology Names: AAA's Recommendation for Common Naming of Advanced Safety Systems*. Tech. rep. 2019. URL: https://www.aaa.com/AAA/common/AAR/files/ADAS-Technology-Names-Research-Report.pdf (Accessed: 7 Dec. 2024).

[Ant+19] André Antakli et al. “Agent-based Web Supported Simulation of Human-robot Collaboration”. In: *Proceedings of the 15th International Conference on Web Information Systems and Technologies - WEBIST*. INSTICC. SciTePress, 2019, pp. 88–99. DOI: 10.5220/0008163000880099.

[Ant+21] André Antakli et al. “HAIL: Modular Agent-Based Pedestrian Imitation Learning”. In: *Advances in Practical Applications of Agents, Multi-Agent Systems, and Social Good. The PAAMS Collection*. Ed. by Frank Dignum, Juan Manuel Corchado, and Fernando De La Prieta. Cham: Springer International Publishing, 2021, pp. 27–39. DOI: 10.1007/978-3-030-85739-4_3.

[ASY14] Paul Atchley, Jing Shi, and Toshiyuki Yamamoto. “Cultural foundations of safety culture: A comparison of traffic safety culture in China, Japan and the United States”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 26 (2014). SI: Traffic safety culture, pp. 317–325. DOI: 10.1016/j.trf.2014.01.004.

[BBU12] Erich B Bergiel, Blaise J Bergiel, and John W Upson. “Revisiting Hofstede’s Dimensions : Examining the Cultural Convergence of the United States and Japan”. In: *American Journal of Management* 12.1 (2012), pp. 69–79. URL: https://www.researchgate.net/publication/284043384_Revisiting_Hofstede's_Dimensions_Examining_the_Cultural_Convergence_of_the_United_States_and_Japan (Accessed: 7 Dec. 2024).

[BE14] Bettina Bartels and Christian Erbsmehl. *Bewegungsverhalten von Fußgängern im Straßenverkehr, Teil 2*. Vol. 268. FAT-Schriftenreihe. Forschungsvereinigung Automobiltechnik E.V., 2014. URL: https://edocs.tib.eu/files/e01fn14/805007342.pdf (Accessed: 7 Dec. 2024).

[Bea+15] Alejandro Beacco et al. “Footstep parameterized motion blending using barycentric coordinates”. In: *Computers Graphics* 47 (2015), pp. 105–112. DOI: 10.1016/j.cag.2014.12.004.

[Ben+15] Samy Bengio et al. “Scheduled sampling for sequence prediction with recurrent Neural networks”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’15. Montreal, Canada: MIT Press, 2015, pp. 1171–1179.

[BFC09] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. “Semantic object classes in video: A high-definition ground truth database”. In: *Pattern Recognition Letters* 30.2 (2009). Video-based Object and Event Analysis, pp. 88–97. DOI: 10.1016/j.patrec.2008.04.005.

[BH00] Matthew Brand and Aaron Hertzmann. “Style machines”. In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’00. USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 183–192. DOI: 10.1145/344779.344865.

[Bha+12] Gaurav Bharaj et al. “Automatically Rigging Multi-component Characters”. In: vol. 31. 2pt4. 2012, pp. 755–764. DOI: 10.1111/j.1467-8659.2012.03034.x.

[Bhu21] Chirag Bhuvaneshwara. “Simulating Fast-Movements with Mixture-of-Expert Models: An Interactive Boxing Controller”. Master thesis. Saarland University, 2021.

[Blo+11] Torsten Blochwitz et al. “The Functional Mockup Interface for Tool independent Exchange of Simulation Models”. In: *Proceedings of the 8th International Modelica Conference*. Vol. 63. Linköping Electronic Conference Proceedings. Linköping University Press, 2011, pp. 105–114. DOI: 10.3384/ecp11063105.

[Boh21] Jan Bohnerth. “Scaling Constraints Instead of Motion - Towards a Scale-Aware Motion Transfer Function in the Motion Simulation Framework MOSIM”. Bachelor thesis. University of Applied Sciences Saarland, 2021.

[Boh23] Jan Bohnerth. “Towards VR Avatar Visualization with the Help of Conditional Variational Autoencoders and Recurrent Neural Networks”. Master thesis. University of Applied Sciences Saarland, 2023.

[Bow16] R Bowden. *Learning Statistical Models of Human Motion*. Hilton Head, South Carolina, U.S.A., 20000715 - 20000716. URL: https://www.researchgate.net/publication/2423255_Learning_Statistical_Models_of_Human_Motion (Accessed: 7 Dec. 2024).

[BPE10] B. J. H. van Basten, P. W. A. M. Peeters, and A. Egges. “The step space: example-based footprint-driven motion synthesis”. In: *Computer Animation and Virtual Worlds* 21.3-4 (2010), pp. 433–441. DOI: 10.1002/cav.342.

[Bra+19] Markus Braun et al. “EuroCity Persons: A Novel Benchmark for Person Detection in Traffic Scenes”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.8 (2019), pp. 1844–1861. DOI: 10.1109/TPAMI.2019.2897684.

[Bro+13] Marilyne Brosseau et al. “The impact of waiting time and other factors on dangerous pedestrian crossings and violations at signalized intersections: A case study in Montreal”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 21 (2013), pp. 159–172. DOI: 10.1016/j.trf.2013.09.010.

[Bro+20] Tom B. Brown et al. “Language models are few-shot learners”. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS ’20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[Bun23] Bundesministerium für Digitales und Verkehr. *Autonome-Fahrzeuge-Genehmigungs-und-Betriebs-Verordnung vom 24. Juni 2022 (BGBl. I S. 986), die durch Artikel 10 der Verordnung vom 20. Juli 2023 (BGBl. 2023 I Nr. 199) geändert worden ist*. Bundesgesetzblatt. BGBl. I S. 986. 2023. URL: https://www.gesetze-im-internet.de/afgbv/ (Accessed: 7 Dec. 2024).

[BV99] Volker Blanz and Thomas Vetter. “A morphable model for the synthesis of 3D faces”. In: *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’99. USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 187–194. DOI: 10.1145/311535.311556.

[Cae+20] Holger Caesar et al. “nuScenes: A Multimodal Dataset for Autonomous Driving”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 11618–11628. DOI: 10.1109/CVPR42600.2020.01164.

[Cai+21] Yujun Cai et al. “A Unified 3D Human Motion Synthesis Model via Conditional Variational Auto-Encoder”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 11625–11635. DOI: 10.1109/ICCV48922.2021.01144.

[Cam+20] Scott Camazine et al. *Self-Organization in Biological Systems*. Princeton University Press, May 2020. DOI: 10.2307/j.ctvzxx9tx.

[CAR15] Victor Cantillo, Julian Arellana, and Manuel Rolong. “Modelling pedestrian crossing behaviour in urban roads: A latent variable approach”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 32 (2015), pp. 56–67. DOI: 10.1016/j.trf.2015.04.008.

[CFG12] Camilo Charron, Aurélie Festoc, and Nicolas Guéguen. "Do child pedestrians deliberately take risks when they are in a hurry? An experimental study on a simulator". In: *Transportation Research Part F: Traffic Psychology and Behaviour* 15.6 (2012), pp. 635–643. DOI: 10.1016/j.trf.2012.07.001.

[CGC08] Allen M. Y. Cheong, Duane R. Geruschat, and Nathan Congdon. "Traffic Gap Judgment in People with Significant Peripheral Field Loss". In: *Optometry and Vision Science* 85.1 (2008). ISSN: 1538-9235. URL: https://journals.lww.com/optvissci/fulltext/2008/01000/traffic_gap_judgment_in_people_with_significant.7.aspx.

[Cha+19] Jacky C. P. Chan et al. "A generic framework for editing and synthesizing multimodal data with relative emotion strength". In: *Computer Animation and Virtual Worlds* 30.6 (2019). e1871 cav.1871, e1871. DOI: 10.1002/cav.1871.

[Che+23] Xuanzhu Chen et al. "Camera and LiDAR Fusion for Urban Scene Reconstruction and Novel View Synthesis via Voxel-Based Neural Radiance Fields". In: *Remote Sensing* 15.18 (2023). DOI: 10.3390/rs15184628.

[Chi+00] Diane Chi et al. "The EMOTE model for effort and shape". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 173–182. DOI: 10.1145/344779.352172.

[Cla16] Simon Clavet. *Motion matching and the road to next-gen animation*. 2016. URL: https://archive.org/details/GDC2016Clavet (Accessed: 7 Dec. 2024).

[Cle+19] Koen de Clercq et al. "External Human-Machine Interfaces on Automated Vehicles: Effects on Pedestrian Crossing Decisions". In: *Human Factors* 61.8 (2019). PMID: 30912985, pp. 1353–1370. DOI: 10.1177/0018720819836343.

[Clo+17] Marie-Soleil Cloutier et al. ""Outta my way!" Individual and environmental correlates of interactions between pedestrians and vehicles during street crossings". In: *Accident Analysis Prevention* 104 (2017), pp. 36–45. DOI: 10.1016/j.aap.2017.04.015.

[Coh+13] Ariel Cohen et al. "Guardrail influence on pedestrian crossing behavior at roundabouts". In: *Accident Analysis Prevention* 59 (2013), pp. 452–458. DOI: 10.1016/j.aap.2013.06.019.

[Com+22] European Commission et al. *Study on new mobility patterns in European cities : final report. Task A, EU wide passenger mobility survey*. Publications Office of the European Union, 2022. DOI: doi/10.2832/728583.

[Com24] European Commission. *Facts and Figures Pedestrians*. Brussels: European Road Safety Observatory, 2024. URL: https://road-safety.transport.ec.europa.eu/document/download/59566b2a-648c-45e5-b2ff-957047feda2a_en?filename=ff_pedestrians_14082024.pdf (Accessed: 1 Dec. 2024).

[Con+98] Marie L Connelly et al. "Child pedestrians' crossing gap thresholds". In: *Accident Analysis Prevention* 30.4 (1998), pp. 443–453. DOI: 10.1016/S0001-4575(97)00109-7.

[Cor+16] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3213–3223. DOI: 10.1109/CVPR.2016.350.

[CP11] Brendan Chan and Ananda Pandy. "Forward Collision Mitigation Systems: A Safety Benefits Analysis for Commercial Vehicles via Hardware-in- the-loop Simulation". In: *SAE Int. J. Commer. Veh.* 4.1 (2011), pp. 222–231. DOI: 10.4271/2011-01-2259.

[Cur+19] Luis A. Curiel-Ramirez et al. "Hardware in the loop framework proposal for a semi-autonomous car architecture in a closed route environment". In: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 13.4 (Dec. 2019), pp. 1647–1658. DOI: 10.1007/s12008-019-00619-x.

[Daa04] Winnie Daamen. *Modelling Passenger Flows in Public Transport Facilities*. Delft University Press Science, 2004. ISBN: 9040725217. URL: https://repository.tudelft.nl/record/uuid:e65fb66c-1e55-4e63-8c49-5199d40f60e1 (Accessed: 7 Dec. 2024).

[Dah+19] Tim Dahmen et al. "Digital reality: a model-based approach to supervised learning from synthetic data". In: *AI Perspectives* 1.1 (2019), p. 2. DOI: 10.1186/s42467-019-0002-0.

[DCO13] Aurélie Dommes, Viola Cavallo, and Jennifer Oxley. "Functional declines as predictors of risky street-crossing decisions in older pedestrians". In: *Accident Analysis Prevention* 59 (2013), pp. 135–143. DOI: 10.1016/j.aap.2013.05.017.

[Deg+22] Bruno Degardin et al. "Generative Adversarial Graph Convolutional Networks for Human Action Synthesis". In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022, pp. 2753–2762. DOI: 10.1109/WACV51458.2022.00281.

[dev21] ONNX Runtime developers. *ONNX Runtime*. https://onnxruntime.ai/. 2021. (Accessed: 7 Dec. 2024).

[Do+23] Tiffany D. Do et al. "VALID: a perceptually validated Virtual Avatar Library for Inclusion and Diversity". In: *Frontiers in Virtual Reality* 4 (2023). DOI: 10.3389/frvir.2023.1248915.

[DOA15] Y.I. Demiroz, Pelin Onelcin, and Yalcin Alver. "Illegal road crossing behavior of pedestrians at overpass locations: Factors affecting gap acceptance, crossing times and overpass use". In: *Accident Analysis Prevention* 80 (2015), pp. 220–228. DOI: 10.1016/j.aap.2015.04.018.

[Dob15] Przemyslaw Dobrowolski. *Swing-twist decomposition in Clifford algebra*. 2015. DOI: 10.48550/arXiv.1506.05481.

[Dom+15] Aurélie Dommes et al. "Red light violations by adult pedestrians and other safety-related behaviors at signalized crosswalks". In: *Accident Analysis Prevention* 80 (2015), pp. 67–75. DOI: 10.1016/j.aap.2015.04.002.

[Don+17] Yuzhu Dong et al. "Adult2Child: dynamic scaling laws to create child-like motion". In: *Proceedings of the 10th International Conference on Motion in Games*. MIG '17. Barcelona, Spain: Association for Computing Machinery, 2017. DOI: 10.1145/3136457.3136460.

[Don+20] Yuzhu Dong et al. "Adult2child: Motion Style Transfer using CycleGANs". In: *Proceedings of the 13th ACM SIGGRAPH Conference on Motion, Interaction and Games*. MIG '20. Virtual Event, SC, USA: Association for Computing Machinery, 2020. DOI: 10.1145/3424636.3426909.

[Dos+17] Alexey Dosovitskiy et al. *CARLA: An Open Urban Driving Simulator*. 2017. arXiv: 1711.03938.

[DPV06] José Mario De Martino, Léo Pini Magalhães, and Fábio Violaro. "Facial animation based on context-dependent visemes". In: *Computers Graphics* 30.6 (2006), pp. 971–980. DOI: https://doi.org/10.1016/j.cag.2006.08.017.

[DT05] Navneet Dalal and Bill Triggs. “Histograms of oriented gradients for human detection”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[DT17] Debargha Dey and Jacques Terken. “Pedestrian Interaction with Vehicles: Roles of Explicit and Implicit Communication”. In: *Proceedings of the 9th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. AutomotiveUI '17. Oldenburg, Germany: Association for Computing Machinery, 2017, pp. 109–113. DOI: 10.1145/3122986.3123009.

[Du+16] Han Du et al. “Scaled functional principal component analysis for human motion synthesis”. In: *Proceedings of the 9th International Conference on Motion in Games*. MIG '16. Burlingame, California: Association for Computing Machinery, 2016, pp. 139–144. DOI: 10.1145/2994258.2994277.

[Du+19] Han Du et al. “Stylistic Locomotion Modeling and Synthesis using Variational Generative Models”. In: *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games*. MIG '19. Newcastle upon Tyne, United Kingdom: Association for Computing Machinery, 2019. DOI: 10.1145/3359566.3360083.

[Dun12] George Dunbar. “The relative risk of nearside accidents is high for the youngest and oldest pedestrians”. In: *Accident Analysis Prevention* 45 (2012), pp. 517–521. DOI: 10.1016/j.aap.2011.09.001.

[Dur+16] Funda Durupinar et al. “PERFORM: Perceptual Approach for Adding OCEAN Personality to Human Motion Using Laban Movement Analysis”. In: *ACM Trans. Graph.* 36.1 (Oct. 2016). DOI: 10.1145/2983620.

[EB17] Rune Elvik and Torkel Bjørnskau. “Safety-in-numbers: A systematic review and meta-analysis of evidence”. In: *Safety Science* 92 (2017), pp. 274–282. DOI: 10.1016/j.ssci.2015.07.017.

[EF78] Paul Ekman and Wallace V Friesen. “Facial action coding system”. In: *Environmental Psychology & Nonverbal Behavior* (1978). DOI: 10.1037/t27734-000.

[EH77] Wolfgang Eberhardt and Gundolf Himbert. *Bewegungsgeschwindigkeiten: Versuchsergebnisse nichtmotorisierter Verkehrsteilnehmer*. German. Vol. 15. Verkehrsunfall 4. Accession Number: 01305168. Bundesanstalt für Straßenwesen (BASt): Information Verlag, Apr. 1977, pp. 79–84.

[ELV07] Andreas Ess, Bastian Leibe, and Luc Van Gool. “Depth and Appearance for Mobile Scene Analysis”. In: *2007 IEEE 11th International Conference on Computer Vision*. 2007, pp. 1–8. DOI: 10.1109/ICCV.2007.4409092.

[EM11] Udo Enste and Wolfgang Mahnke. In: *at - Automatisierungstechnik* 59.7 (2011), pp. 397–404. DOI: doi:10.1524/auto.2011.0934.

[EM19] Fatima El Jamiy and Ronald Marsh. “Survey on depth perception in head mounted displays: distance estimation in virtual reality, augmented reality, and mixed reality”. In: *IET Image Processing* 13.5 (2019), pp. 707–712. DOI: 10.1049/iet-ipr.2018.5920.

[Eri25] Erik Herrmann. “Exploration Algorithms and Data Management for the Application of Motion Synthesis in Multi-Agent Simulations”. In preparation. PhD thesis. Saarland University, 2025.

[Eur17] Euro NCAP. *Euro NCAP 2025 Roadmap: In pursuit of Vision Zero*. Tech. rep. 2017. URL: https://cdn.euroncap.com/media/30700/euroncap-roadmap-2025-v4.pdf (Accessed: 1 Dec. 2024).

[Eur23] Euro NCAP. *European New Car Assessment Programme (Euro NCAP) - Test Protocol AEB/LSS VRU Systems Version 4.4*. Tech. rep. 2023. URL: https://cdn.euroncap.com/media/77299/euro-ncap-aeb-lss-vru-test-protocol-v44.pdf (Accessed: 1 Dec. 2024).

[Eur24] Euro NCAP. *Assisted Driving: Highway & Interurban Assist Systems - Test & Assessment Protocol (Version 2.1)*. Tech. rep. 2024. URL: https://www.euroncap.com/media/80151/euro-ncap-ad-test-and-assessment-protocol-v21.pdf (Accessed: 1 Dec. 2024).

[Fab+22] Simone Fabbrizzi et al. "A survey on bias in visual datasets". In: *Computer Vision and Image Understanding* 223 (2022), p. 103552. DOI: https://doi.org/10.1016/j.cviu.2022.103552.

[Far+21] Abolfazl Farahani et al. "A Brief Review of Domain Adaptation". In: *Advances in Data Science and Information Engineering*. Springer International Publishing, 2021, pp. 877–894. DOI: 10.1007/978-3-030-71704-9_65.

[Fel+18] Ilja T. Feldstein et al. "Pedestrian simulators for traffic research: state of the art and future of a motion lab". In: *International Journal of Human Factors Modelling and Simulation* 6.4 (2018). DOI: 10.1504/ijhfms.2018.096128.

[Fen+14] Andrew Feng et al. "Fast, automatic character animation pipelines". In: *Computer Animation and Virtual Worlds* 25.1 (2014), pp. 3–16. DOI: 10.1002/cav.1560.

[Fie+20] Mihai Fieraru et al. "Three-Dimensional Reconstruction of Human Interactions". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 7212–7221. DOI: 10.1109/CVPR42600.2020.00724.

[FKK10] Jolyon J. Faria, Stefan Krause, and Jens Krause. "Collective behavior in road crossing pedestrians: the role of social information". In: *Behavioral Ecology* 21.6 (Sept. 2010), pp. 1236–1242. DOI: 10.1093/beheco/arq141.

[FKK20] Ilja T. Feldstein, Felix M. Kölsch, and Robert Konrad. "Egocentric Distance Perception: A Comparative Study Investigating Differences Between Real and Virtual Environments". In: *Perception* 49.9 (2020), pp. 940–967. DOI: 10.1177/0301006620951997.

[Fra+12] Katerina Fragkiadaki et al. "Two-Granularity Tracking: Mediating Trajectory and Detection Graphs for Tracking under Occlusions". In: *Computer Vision – ECCV 2012*. Ed. by Andrew Fitzgibbon et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 552–565. DOI: 10.1007/978-3-642-33715-4_40.

[Fra+15] Katerina Fragkiadaki et al. "Recurrent Network Models for Human Dynamics". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 4346–4354. DOI: 10.1109/ICCV.2015.494.

[FSS03] Klaus Fischer, Michael Schillo, and Jörg Siekmann. "Holonic Multiagent Systems: A Foundation for the Organisation of Multiagent Systems". In: *Holonic and Multi-Agent Systems for Manufacturing*. Ed. by Vladimír Mařík, Duncan McFarlane, and Paul Valckenaers. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 71–80. ISBN: 978-3-540-45185-3.

[Gai+16] Adrien Gaidon et al. "Virtual Worlds as Proxy for Multi-object Tracking Analysis". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 4340–4349. DOI: 10.1109/CVPR.2016.470.

[Gai+18] Felix Gaisbauer et al. “Presenting a Deep Motion Blending Approach for Simulating Natural Reach Motions”. In: *EG 2018 - Posters*. Ed. by Eakta Jain and Jirí Kosinka. The Eurographics Association, 2018. DOI: 10.2312/egp.20181010.

[Gai+19] Felix Gaisbauer et al. “Natural Posture Blending Using Deep Neural Networks”. In: *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games*. MIG ’19. Newcastle upon Tyne, United Kingdom: Association for Computing Machinery, 2019. DOI: 10.1145/3359566.3360052.

[Gai22] Gaisbauer, Felix. “Development of a Modular Digital Human Simulation Framework for Planning of Manual Assembly Processes”. PhD thesis. Ulm: Ulm University, Apr. 2022. URL: https://oparu.uni-ulm.de/items/af5abadd-fee3-4a38-8eca-0d754db0ed5b (Accessed: 1 Dec. 2024).

[Gar+15] Rebeca I. García-Betances et al. “Using Virtual Reality for Cognitive Training of the Elderly”. In: *American Journal of Alzheimer’s Disease & Other Dementias®* 30.1 (2015). PMID: 25107931, pp. 49–54. DOI: 10.1177/1533317514545866.

[Ger+06] Duane R. Geruschat et al. “Gaze Behavior of the Visually Impaired During Street Crossing”. In: *Optometry and Vision Science* 83.8 (2006). ISSN: 1538-9235. URL: https://journals.lww.com/optvissci/fulltext/2006/08000/gaze_behavior_of_the_visually_impaired_during.8.aspx.

[GG23] Roberto Gozalo-Brizuela and Eduardo C. Garrido-Merchan. *ChatGPT is not all you need. A State of the Art Review of large Generative AI models*. 2023. arXiv: 2301.04655 [cs.LG].

[Gho+23] Anindita Ghosh et al. “IMoS: Intent-Driven Full-Body Motion Synthesis for Human-Object Interactions”. In: *Computer Graphics Forum* 42.2 (2023), pp. 1–12. DOI: 10.1111/cgf.14739.

[GHÖ14] Tina Gehlert, Carmen Hagemeister, and Türker Özkan. “Traffic safety climate attitudes of road users in Germany”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 26 (2014). SI: Traffic safety culture, pp. 326–336. DOI: 10.1016/j.trf.2013.12.011.

[GHT03] Duane R. Geruschat, Shirin E. Hassan, and Kathleen A. Turano. “Gaze Behavior while Crossing Complex Intersections”. In: *Optometry and Vision Science* 80.7 (July 2003), pp. 515–528. DOI: 10.1097/00006324-200307000-00013.

[Gil+92] N. Giladi et al. “Motor blocks in Parkinson’s disease”. In: *Neurology* 42.2 (1992), pp. 333–333. DOI: 10.1212/WNL.42.2.333.

[Gle98] Michael Gleicher. “Retargetting motion to new characters”. In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’98. New York, NY, USA: Association for Computing Machinery, 1998, pp. 33–42. DOI: 10.1145/280814.280820.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for autonomous driving? The KITTI vision benchmark suite”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.

[Gol90] Lewis R. Goldberg. “An Alternative "Description of Personality": The Big-Five Factor Structure”. In: *Journal of Personality and Social Psychology* 59.6 (1990), pp. 1216–1229. DOI: 10.1037/0022-3514.59.6.1216.

[Goo+20] Ian Goodfellow et al. “Generative adversarial networks”. In: *Commun. ACM* 63.11 (Oct. 2020), pp. 139–144. DOI: 10.1145/3422622.

[Got+18] Paulo Gotardo et al. “Practical dynamic facial appearance modeling and acquisition”. In: *ACM Trans. Graph.* 37.6 (Dec. 2018). DOI: 10.1145/3272127.3275073.

[Gov60] Government of Japan. *Road Traffic Act, Act No. 105 of June 25, 1960*. June 1960. URL: https://www.japaneselawtranslation.go.jp/en/laws/view/2962 (Accessed: 21 July 2024).

[GP12] Thomas Geijtenbeek and Nicolas Pronost. “Interactive Character Animation Using Simulated Physics: A State-of-the-Art Review”. In: *Computer Graphics Forum* 31.8 (2012), pp. 2492–2515. DOI: 10.1111/j.1467-8659.2012.03189.x.

[Gra07] Marie-Axelle Granié. “Gender differences in preschool children’s declared and behavioral compliance with pedestrian rules”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 10.5 (2007), pp. 371–382. DOI: 10.1016/j.trf.2007.02.002.

[Guo+15] Shihui Guo et al. “Adaptive motion synthesis for virtual characters: a survey”. In: *The Visual Computer* 31.5 (May 2015), pp. 497–512. DOI: 10.1007/s00371-014-0943-4.

[Hab+17] Ikhsanul Habibie et al. “A Recurrent Variational Autoencoder for Human Motion Synthesis”. In: *Proceedings of the British Machine Vision Conference (BMVC)*. British Machine Vision Association, 2017, pp. 119.1–119.12. DOI: 10.5244/C.31.119.

[Häm+14] Perttu Hämäläinen et al. “Online motion synthesis using sequential Monte Carlo”. In: *ACM Trans. Graph.* 33.4 (July 2014). DOI: 10.1145/2601097.2601218.

[Hec+08] Chris Hecker et al. “Real-time motion retargeting to highly varied user-created morphologies”. In: *ACM Trans. Graph.* 27.3 (Aug. 2008), pp. 1–11. DOI: 10.1145/1360612.1360626.

[Hel+21] Lorena Hell et al. “Pedestrian Behavior in Japan and Germany: A Review”. In: *2021 IEEE Intelligent Vehicles Symposium (IV)*. 2021, pp. 1529–1536. DOI: 10.1109/IV48863.2021.9575809.

[Her+18] Nico Herbig et al. “How machine perception relates to human perception: visual saliency and distance in a frame-by-frame semantic segmentation task for highly/fully automated driving”. In: *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*. SEFAIS ’18. Gothenburg, Sweden: Association for Computing Machinery, 2018, pp. 6–10. DOI: 10.1145/3194085.3194092.

[Her+19a] Erik Herrmann et al. “Adaptive gaussian mixture trajectory model for physical model control using motion capture data”. In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D ’19. Montreal, Quebec, Canada: Association for Computing Machinery, 2019. DOI: 10.1145/3306131.3317027.

[Her+19b] Erik Herrmann et al. “Motion Data and Model Management for Applied Statistical Motion Synthesis”. In: *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*. Ed. by Marco Agus, Massimiliano Corsini, and Ruggero Pintus. The Eurographics Association, 2019. DOI: 10.2312/stag.20191366.

[HH10] Carol Holland and Ros Hill. “Gender differences in factors predicting unsafe crossing decisions in adult pedestrians across the lifespan: A simulation study”. In: *Accident Analysis Prevention* 42.4 (2010), pp. 1097–1106. DOI: 10.1016/j.aap.2009.12.023.

[HHM05] Geert Hofstede, Gert Jan Hofstede, and Michael Minkov. *Cultures and Organizations: Software of the Mind*. 2nd ed. New York: McGrawHill, 2005. ISBN: 0070293074.

[HK09] Alexis Heloir and Michael Kipp. "EMBR – A Realtime Animation Engine for Interactive Embodied Agents". In: *Intelligent Virtual Agents*. Ed. by Zsófia Ruttkay et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 393–404. ISBN: 978-3-642-04380-2.

[HK10] Alexis Heloir and Michael Kipp. "Real-time animation of interactive agents: Specification and realization". In: *Applied Artificial Intelligence* 24.6 (2010), pp. 510–529. DOI: 10.1080/08839514.2010.492161.

[HKS17] Daniel Holden, Taku Komura, and Jun Saito. "Phase-functioned neural networks for character control". In: *ACM Trans. Graph.* 36.4 (July 2017). DOI: 10.1145/3072959.3073663.

[Hof01] Geert Hofstede. *Culture's consequences: Comparing values, behaviors, institutions and organizations across nations*. 2nd ed. Thousand Oaks, CA: Sage, 2001.

[Hof11] Geert Hofstede. "Dimensionalizing Cultures: The Hofstede Model in Context". In: *Online Readings in Psychology and Culture* 2.1 (2011), pp. 1–26. DOI: 10.9707/2307-0919.1014.

[Hol+15] Daniel Holden et al. "Learning motion manifolds with convolutional autoencoders". In: *SIGGRAPH Asia 2015 Technical Briefs*. SA '15. Kobe, Japan: Association for Computing Machinery, 2015. DOI: 10.1145/2820903.2820918.

[Hol+17] Daniel Holden et al. "Fast Neural Style Transfer for Motion Data". In: *IEEE Computer Graphics and Applications* 37.4 (2017), pp. 42–49. DOI: 10.1109/MCG.2017.3271464.

[Hol+20] Daniel Holden et al. "Learned motion matching". In: *ACM Trans. Graph.* 39.4 (Aug. 2020). DOI: 10.1145/3386569.3392440.

[Hol18] Daniel Holden. "Character control with neural networks and machine learning". In: *Proc. of GDC 2018* 1 (2018), p. 2. URL: https://www.gdcvault.com/play/1025389/Character-Control-with-Neural-Networks (Accessed: 1 Dec. 2024).

[HPP05] Eugene Hsu, Kari Pulli, and Jovan Popović. "Style translation for human motion". In: vol. 24. 3. New York, NY, USA: Association for Computing Machinery, July 2005, pp. 1082–1089. DOI: 10.1145/1073204.1073315.

[HS12] Shirin E. Hassan and Benjamin D. Snyder. "Street-Crossing Decision-Making: A Comparison between Patients with Age-Related Macular Degeneration and Normal Vision". In: *Investigative Ophthalmology Visual Science* 53.10 (Sept. 2012), pp. 6137–6144. DOI: 10.1167/iovs.12-10023.

[HSK16] Daniel Holden, Jun Saito, and Taku Komura. "A deep learning framework for character motion synthesis and editing". In: *ACM Trans. Graph.* 35.4 (July 2016). DOI: 10.1145/2897824.2925975.

[Hu+24] Lei Hu et al. "Pose-Aware Attention Network for Flexible Motion Retargeting by Body Part". In: *IEEE Transactions on Visualization and Computer Graphics* 30.8 (2024), pp. 4792–4808. DOI: 10.1109/TVCG.2023.3277918.

[Hwa+15] Soonmin Hwang et al. "Multispectral pedestrian detection: Benchmark dataset and baseline". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1037–1045. DOI: 10.1109/CVPR.2015.7298706.

[IA17] Miho Iryo-Asano and Wael K.M. Alhajyaseen. "Modeling pedestrian crossing speed profiles considering speed change behavior for the safety assessment of signalized intersections". In: *Accident Analysis Prevention* 108 (2017), pp. 332–342. DOI: 10.1016/j.aap.2017.08.028.

[IBM17] IBM-Corp. *IBM SPSS Statistics for Windows*. Armonk, NY: IBM Corp, 2017. URL: https://www.ibm.com/de-de/analytics/spss-statistics-software (Accessed: 1 Dec. 2024).

[IN08] Muhammad Moazzam Ishaque and Robert B. Noland. "Behavioural Issues in Pedestrian Speed Choice and Street Crossing Behaviour: A Review". In: *Transport Reviews* 28.1 (2008), pp. 61–85. DOI: 10.1080/01441640701365239.

[Ino+10] Shigeru Inoue et al. "Association between Perceived Neighborhood Environment and Walking among Adults in 4 Cities in Japan". In: *Journal of Epidemiology* 20.4 (2010), pp. 277–286. DOI: 10.2188/jea.JE20090120.

[Ins24] Hofstede Insights. *Country Comparison Tool: Germany, Japan*. 2024. URL: https://www.hofstede-insights.com/country-comparison-tool?countries=germany,japan (Accessed: 21 July 2024).

[Int10] International Organization for Standardization. *Basic Human Body Measurements for Technological Design – Part 2: Statistical Summaries of Body Measurements from Individual ISO Populations*. Standard ISO 7250-2:2010. Geneva, Switzerland: International Organization for Standardization, 2010. URL: https://www.iso.org/standard/52060.html (Accessed: 21 July 2024).

[Jay+20] Mathilde Jay et al. "The light is red: Uncertainty behaviours displayed by pedestrians during illegal road crossing". In: *Accident Analysis Prevention* 135 (2020), p. 105369. DOI: 10.1016/j.aap.2019.105369.

[Jia+16] Yuanyuan Jiang et al. "Acting together: Joint pedestrian road crossing in an immersive virtual environment". In: *2016 IEEE Virtual Reality (VR)*. 2016, pp. 193–194. DOI: 10.1109/VR.2016.7504719.

[Joh73] Gunnar Johansson. "Visual perception of biological motion and a model for its analysis". In: *Perception & Psychophysics* 14.2 (June 1973), pp. 201–211. DOI: 10.3758/BF03212378.

[Jul+20] Arthur Juliani et al. *Unity: A general platform for intelligent agents*. 2020. arXiv: 1809.02627.

[KB17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[KC77] Lynn T. Kozlowski and James E. Cutting. "Recognizing the sex of a walker from a dynamic point-light display". In: *Perception & Psychophysics* 21.6 (Nov. 1977), pp. 575–580. DOI: 10.3758/BF03198740.

[Ken12] Ben Kenwright. "Inverse Kinematics – Cyclic Coordinate Descent (CCD)". In: *Journal of Graphics Tools* 16.4 (2012), pp. 177–217. DOI: 10.1080/2165347X.2013.823362.

[KG14] Christoph G. Keller and Dariu M. Gavrila. "Will the Pedestrian Cross? A Study on Pedestrian Path Prediction". In: *IEEE Transactions on Intelligent Transportation Systems* 15.2 (2014), pp. 494–506. DOI: 10.1109/TITS.2013.2280766.

[KGP02] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. "Motion graphs". In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '02. San Antonio, Texas: Association for Computing Machinery, 2002, pp. 473–482. DOI: 10.1145/566570.566605.

[KHW19] Lars Kooijman, Riender Happee, and Joost C. F. de Winter. "How Do eHMIs Affect Pedestrians' Crossing Behavior? A Study Using a Head-Mounted Display Combined with a Motion Suit". In: *Information* 10.12 (2019). DOI: 10.3390/info10120386.

[Kim14] Jong-Wook Kim. “Online Joint Trajectory Generation of Human-like Biped Walking”. In: *International Journal of Advanced Robotic Systems* 11.2 (2014), p. 19. DOI: 10.5772/57415.

[Kle+19] Benedikt Kleinmeier et al. “Vadere: An Open-Source Simulation Framework to Promote Interdisciplinary Understanding”. In: *Collective Dynamics* 4 (Jan. 1, 2019). DOI: 10.17815/CD.2019.21. published.

[KLV20] Taesoo Kwon, Yoonsang Lee, and Michiel Van De Panne. “Fast and flexible multilegged locomotion using learned centroidal dynamics”. In: *ACM Trans. Graph.* 39.4 (Aug. 2020). DOI: 10.1145/3386569.3392432.

[Kop+06] Stefan Kopp et al. “Towards a Common Framework for Multimodal Generation: The Behavior Markup Language”. In: *Intelligent Virtual Agents*. Ed. by Jonathan Gratch et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 205–217. ISBN: 978-3-540-37594-4.

[KR91] Randall R. Kleinhesselink and Eugene A. Rosa. “Cognitive Representation of Risk Perceptions: A Comparison of Japan and the United States”. In: *Journal of Cross-Cultural Psychology* 22.1 (1991), pp. 11–28. DOI: 10.1177/0022022191221004.

[Kra+12] Daniel Krajzewicz et al. “Recent Development and Applications of SUMO - Simulation of Urban MObility”. In: *International Journal On Advances in Systems and Measurements* 5.3 (Dec. 2012), pp. 128–138. ISSN: 1942-261X. URL: https://elib.dlr.de/80483/ (Accessed: 21 July 2024).

[KRO18] Semyon Kalantarov, Raziel Riemer, and Tal Oron-Gilad. “Pedestrians’ road crossing decisions and body parts’ movements”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 53 (2018), pp. 155–171. DOI: 10.1016/j.trf.2017.09.012.

[KRT20] Iuliia Kotseruba, Amir Rasouli, and John K. Tsotsos. “Do They Want to Cross? Understanding Pedestrian Intention for Behavior Prediction”. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 1688–1693. DOI: 10.1109/IV47402.2020.9304591.

[Kuk+01] Robert Kukla et al. “PEDFLOW: Development of an Autonomous Agent Model of Pedestrian Flow”. In: *Transportation Research Record* 1774.1 (2001), pp. 11–17. DOI: 10.3141/1774-02.

[KW22] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML].

[Kwi+22] Ariel Kwiatkowski et al. “A Survey on Reinforcement Learning Methods in Character Animation”. In: *Computer Graphics Forum* 41.2 (2022), pp. 613–639. DOI: 10.1111/cgf.14504.

[Lan+11] Florian Lange et al. “The dark side of stimulus control—Associations between contradictory stimulus configurations and pedestrians’ and cyclists’ illegal street crossing behavior”. In: *Accident Analysis Prevention* 43.6 (2011), pp. 2166–2172. DOI: 10.1016/j.aap.2011.06.008.

[Lan+16] Florian Lange et al. “Road crossing behavior under traffic light conflict: Modulating effects of green light duration and signal congruency”. In: *Accident Analysis Prevention* 95 (2016), pp. 292–298. DOI: 10.1016/j.aap.2016.07.023.

[LBC13] Régis Lobjois, Nicolas Benguigui, and Viola Cavallo. “The effects of age and traffic density on street-crossing behavior”. In: *Accident Analysis Prevention* 53 (2013), pp. 166–175. DOI: 10.1016/j.aap.2012.12.028.

[LC07] Régis Lobjois and Viola Cavallo. "Age-related differences in street-crossing decisions: The effects of vehicle speed and time constraints on gap selection in an estimation task". In: *Accident Analysis Prevention* 39.5 (2007), pp. 934–943. DOI: 10.1016/j.aap.2006.12.013.

[LC09] Régis Lobjois and Viola Cavallo. "The effects of aging on street-crossing behavior: From estimation to actual crossing". In: *Accident Analysis Prevention* 41.2 (2009), pp. 259–267. DOI: 10.1016/j.aap.2008.12.001.

[LCF00] J. P. Lewis, Matt Cordner, and Nickson Fong. "Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 165–172. DOI: 10.1145/344779.344862.

[Leb76] Takie Sigiyama Lebra. *Japanese Pataterns of Behaviour*. East-West Center Books. Honolulu: University of Hawaii Press, 1976. DOI: 10.1515/9780824846404.

[Lee+02] Jehee Lee et al. "Interactive control of avatars animated with human motion data". In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '02. San Antonio, Texas: Association for Computing Machinery, 2002, pp. 491–500. DOI: 10.1145/566570.566607.

[Lee+10] Yongjoon Lee et al. "Motion fields for interactive character locomotion". In: *ACM Trans. Graph.* 29.6 (Dec. 2010). DOI: 10.1145/1882261.1866160.

[Lev60] Herbert Levene. "Robust Tests for Equality of Variances". In: (1960). Ed. by Ira Olkin, pp. 278–292.

[LGN14] Andreas M. Lehrmann, Peter V. Gehler, and Sebastian Nowozin. "Efficient Nonlinear Markov Models for Human Motion". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1314–1321. DOI: 10.1109/CVPR.2014.171.

[LH19] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG].

[Li+17] Xiaofei Li et al. "A Unified Framework for Concurrent Pedestrian and Cyclist Detection". In: *IEEE Transactions on Intelligent Transportation Systems* 18.2 (2017), pp. 269–281. DOI: 10.1109/TITS.2016.2567418.

[Li+18] Zimo Li et al. *Auto-Conditioned Recurrent Networks for Extended Complex Human Motion Synthesis*. 2018. arXiv: 1707.05363 [cs.LG].

[Li+19] Yanran Li et al. "Efficient convolutional hierarchical autoencoder for human motion prediction". In: *The Visual Computer* 35.6 (June 2019), pp. 1143–1156. DOI: 10.1007/s00371-019-01692-9.

[Li+21] Ruilong Li et al. "AI Choreographer: Music Conditioned 3D Dance Generation with AIST++". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 13381–13392. DOI: 10.1109/ICCV48922.2021.01315.

[Lia+22] Zhouyingcheng Liao et al. "Skeleton-Free Pose Transfer for Stylized 3D Characters". In: *Computer Vision – ECCV 2022*. Ed. by Shai Avidan et al. Cham: Springer Nature Switzerland, 2022, pp. 640–656. ISBN: 978-3-031-20086-1.

[Lin+20] Hung Yu Ling et al. "Character controllers using motion VAEs". In: *ACM Trans. Graph.* 39.4 (Aug. 2020). DOI: 10.1145/3386569.3392422.

[Lip+13] Krsto Lipovac et al. “The influence of a pedestrian countdown display on pedestrian behavior at signalized pedestrian crossings”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 20 (2013), pp. 121–134. DOI: 10.1016/j.trf.2013.07.002.

[LKL10] Yoonsang Lee, Sungeun Kim, and Jehee Lee. “Data-driven biped control”. In: *ACM Trans. Graph.* 29.4 (July 2010). DOI: 10.1145/1778765.1781155.

[LLL18] Kyungho Lee, Seyoung Lee, and Jehee Lee. “Interactive character animation by learning multi-objective control”. In: *ACM Trans. Graph.* 37.6 (Dec. 2018). DOI: 10.1145/3272127.3275071.

[LLZ23] Zhuopeng Li, Lu Li, and Jianke Zhu. “READ: Large-Scale Neural Scene Rendering for Autonomous Driving”. In: vol. 37. 2. June 2023, pp. 1522–1529. DOI: 10.1609/aaai.v37i2.25238.

[LM07] Neil D. Lawrence and Andrew J. Moore. “Hierarchical Gaussian process latent variable models”. In: *Proceedings of the 24th International Conference on Machine Learning*. ICML ’07. Corvalis, Oregon, USA: Association for Computing Machinery, 2007, pp. 481–488. DOI: 10.1145/1273496.1273557.

[Lop+15] Matthew Loper et al. “SMPL: a skinned multi-person linear model”. In: vol. 34. 6. New York, NY, USA: Association for Computing Machinery, Oct. 2015. DOI: 10.1145/2816795.2818013.

[Ma+19] Yuexin Ma et al. “TrafficPredict: Trajectory Prediction for Heterogeneous Traffic-Agents”. In: vol. 33. 01. July 2019, pp. 6120–6127. DOI: 10.1609/aaai.v33i01.33016120.

[Mah+19] Naureen Mahmood et al. “AMASS: Archive of Motion Capture As Surface Shapes”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 5441–5450. DOI: 10.1109/ICCV.2019.00554.

[Mas+18] I. Mason et al. “Few-shot Learning of Homogeneous Human Locomotion Styles”. In: *Computer Graphics Forum* 37.7 (2018), pp. 143–153. DOI: 10.1111/cgf.13555.

[Mat21] Mathworks. *Simulink - Simulation and Model-Based Design - MATLAB & Simulink*. 2021.

[MC03] R. Bruce Money and John C. Crotts. “The effect of uncertainty avoidance on information search, planning, and purchases of international travel vacations”. In: *Tourism Management* 24.2 (2003), pp. 191–202. DOI: 10.1016/S0261-5177(02)00057-2.

[MC12] Jianyuan Min and Jinxiang Chai. “Motion graphs++: a compact generative model for semantic motion analysis and synthesis”. In: *ACM Trans. Graph.* 31.6 (Nov. 2012). DOI: 10.1145/2366145.2366172.

[MCC09] Jianyuan Min, Yen-Lin Chen, and Jinxiang Chai. “Interactive generation of human animation with deformable motion models”. In: *ACM Trans. Graph.* 29.1 (Dec. 2009). DOI: 10.1145/1640443.1640452.

[McD+09] Rachel McDonnell et al. “Evaluating the effect of motion and body shape on the perceived sex of virtual characters”. In: *ACM Trans. Appl. Percept.* 5.4 (Feb. 2009). DOI: 10.1145/1462048.1462051.

[Meh+20] Dushyant Mehta et al. “XNect: real-time multi-person 3D motion capture with a single RGB camera”. In: *ACM Transactions on Graphics* 39.4 (Aug. 2020). DOI: 10.1145/3386569.3392410.

[Men+22] Qianhui Men et al. “GAN-based reactive motion synthesis with class-aware discriminators for human–human interaction”. In: *Computers Graphics* 102 (2022), pp. 634–645. DOI: 10.1016/j.cag.2021.09.014.

[MHM18] Yuichiro Motegi, Yuma Hijioka, and Makoto Murakami. “Human motion generative model using variational autoencoder”. In: *International Journal of Modeling and Optimization* 8.1 (2018), pp. 8–12. DOI: 10.7763/IJMO.2018.V8.616.

[MK91] Hazel Rose Markus and Shinobu Kitayama. “Culture and the self: Implications for cognition, emotion, and motivation”. In: *Psychological review* 98.2 (1991), p. 224. DOI: 10.1037//0033-295x.98.2.224.

[MLC10] Jianyuan Min, Huajun Liu, and Jinxiang Chai. “Synthesis and editing of personalized stylistic human motion”. In: *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D ’10. Washington, D.C.: Association for Computing Machinery, 2010, pp. 39–46. DOI: 10.1145/1730804.1730811.

[MM04] Michael Meredith and Steve Maddock. “Real-Time Inverse Kinematics : The Return of the Jacobian”. In: *Dept of Computer Science University of Sheffield UK* (2004). URL: https://www.researchgate.net/publication/228980657_Real-time_inverse_kinematics_The_return_of_the_Jacobian (Accessed: 7 Dec. 2024).

[Mon+00] Jean-Sébastien Monzani et al. “Using an Intermediate Skeleton and Inverse Kinematics for Motion Retargeting”. In: *Computer Graphics Forum* 19.3 (2000), pp. 11–19. DOI: 10.1111/1467-8659.00393.

[MPO13] Anat Meir, Yisrael Parmet, and Tal Oron-Gilad. “Towards understanding child-pedestrians’ hazard perception abilities in a mixed reality dynamic environment”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 20 (2013), pp. 90–107. DOI: 10.1016/j.trf.2013.05.004.

[MSC22] Jerad Moxley, Joseph Sharit, and Sara J Czaja. “The Factors Influencing Older Adults’ Decisions Surrounding Adoption of Technology: Quantitative Experimental Study”. In: *JMIR Aging* 5.4 (Nov. 2022), e39890. DOI: 10.2196/39890.

[MSS48] Harold B Maynard, Gustave James Stegemerten, and John L Schwab. *Methods-time measurement*. McGraw-Hill, 1948.

[MTH22] Franc Mihalič, Mitja Truntič, and Alenka Hren. “Hardware-in-the-Loop Simulations: A Historical Overview of Engineering Challenges”. In: *Electronics* 11.15 (2022). DOI: 10.3390/electronics11152462.

[Mus+21] Nora Muscholl et al. “EMIDAS: explainable social interaction-based pedestrian intention detection across street”. In: *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. SAC ’21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, pp. 107–115. DOI: 10.1145/3412841.3441891.

[Nak87] Yoshio Nakashima. “Color limits in the chromaticity diagram for the traffic light signals and its apparent hue measure by color-naming method”. In: *International Association of Traffic and Safety Sciences Review* 13.1 (1987), pp. 54–64.

[Nat09] National Police Agency. *Traffic Regulation Standards*. Tech. rep. 2009, 120 et sqq. URL: https://www.npa.go.jp/laws/notification/koutuu/kisei/kisei20170424.pdf (Accessed: 7 Dec. 2024).

[Nei+10] Mark B. Neider et al. “Pedestrians, vehicles, and cell phones”. In: *Accident Analysis Prevention* 42.2 (2010), pp. 589–594. DOI: 10.1016/j.aap.2009.10.004.

[Nei+11] Mark B. Neider et al. “Walking and talking: Dual-task effects on street crossing behavior in older adults.” In: *Psychology and Aging* 26.2 (June 2011), pp. 260–268. DOI: 10.1037/a0021566.

[NIO19] Hideki Nakamura, Miho Iryo-Asano, and Takashi Oguchi. “Japan”. In: *Global Practices on Road Traffic Signal Control*. Elsevier, 2019, pp. 163–184. DOI: 10.1016/B978-0-12-815302-4.00010-8.

[Nos+18] Cheema Noshaba et al. “Dilated Temporal Fully-Convolutional Network for Semantic Segmentation of Motion Capture Data”. In: *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation - Posters*. Ed. by Melina Skouras. The Eurographics Association, 2018. DOI: 10.2312/sca.20181185.

[OA15] Pelin Onelcin and Yalcin Alver. “Illegal crossing behavior of pedestrians at signalized intersections: Factors affecting the gap acceptance”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 31 (2015), pp. 124–132. DOI: 10.1016/j.trf.2015.04.007.

[OCo19] Siobhan O’Connor. *Virtual Reality and Avatars in Health care*. PMID: 31064283. 2019. DOI: 10.1177/1054773819845824.

[Oxl+05] Jennifer A. Oxley et al. “Crossing roads safely: An experimental study of age differences in gap selection by pedestrians”. In: *Accident Analysis Prevention* 37.5 (2005), pp. 962–971. DOI: 10.1016/j.aap.2005.04.017.

[Oxl+97] Jennie Oxley et al. “Differences in traffic judgements between young and old adult pedestrians”. In: *Accident Analysis Prevention* 29.6 (1997), pp. 839–847. DOI: 10.1016/S0001-4575(97)00053-5.

[Pal+21] Prashant Pala et al. “Analysis of Street-Crossing Behavior: Comparing a CAVE Simulator and a Head-Mounted Display among Younger and Older Adults”. In: *Accident Analysis Prevention* 152 (2021), p. 106004. DOI: https://doi.org/10.1016/j.aap.2021.106004.

[Pap12] Eleonora Papadimitriou. “Theory and models of pedestrian crossing behaviour along urban trips”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 15.1 (2012), pp. 75–94. DOI: 10.1016/j.trf.2011.11.007.

[Par+21] Devi Parikh et al. “Relative Attributes for Enhanced Human-Machine Communication”. In: vol. 26. 1. Sept. 2021, pp. 2153–2159. DOI: 10.1609/aaai.v26i1.8443.

[Pas+19] Adam Paszke et al. “PyTorch: an imperative style, high-performance deep learning library”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[PBV21] Mathis Petrovich, Michael J. Black, and Gül Varol. “Action-Conditioned 3D Human Motion Synthesis with Transformer VAE”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 10965–10975. DOI: 10.1109/ICCV48922.2021.01080.

[PC22] The European Parliament and the Council. “Regulation (EU) 2019/2144 of the European Parliament and of the Council of 27 November 2019 on type-approval requirements for motor vehicles and their trailers, and systems, components and separate technical units intended for such vehicles, as regards their general safety and the protection of vehicle occupants and vulnerable road users.” In: *Official Journal of the European Union* L 352 (2022), pp. 1–40. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32019R2144 (Accessed: 1 Dec. 2024).

[PDS19a] Marie Pelé, Jean-Louis Deneubourg, and Cédric Sueur. "Decision-Making Processes Underlying Pedestrian Behaviors at Signalized Crossing: Part 1. The First to Step off the Kerb". In: *Safety* 5.4 (2019). DOI: 10.3390/safety5040079.

[PDS19b] Marie Pelé, Jean-Louis Deneubourg, and Cédric Sueur. "Decision-Making Processes Underlying Pedestrian Behaviors at Signalized Crossings: Part 2. Do Pedestrians Show Cultural Herding Behavior?" In: *Safety* 5.4 (2019). DOI: 10.3390/safety5040082.

[Ped+11] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *J. Mach. Learn. Res.* (Nov. 2011), pp. 2825–2830. ISSN: 1532-4435.

[Pei09] Jonathan W. Peirce. "Generating stimuli for neuroscience using PsychoPy". In: *Frontiers in Neuroinformatics* 2 (2009). DOI: 10.3389/neuro.11.010.2008.

[Pel+09] S. Pellegrini et al. "You'll never walk alone: Modeling social behavior for multi-target tracking". In: *2009 IEEE 12th International Conference on Computer Vision*. 2009, pp. 261–268. DOI: 10.1109/ICCV.2009.5459260.

[Pel+17] Marie Pelé et al. "Cultural influence of social information use in pedestrian road-crossing behaviours". In: *Royal Society Open Science* 4.2 (2017), p. 160739. DOI: 10.1098/rsos.160739.

[Pen+17] Xue Bin Peng et al. "DeepLoco: dynamic locomotion skills using hierarchical deep reinforcement learning". In: *ACM Trans. Graph.* 36.4 (July 2017). DOI: 10.1145/3072959.3073602.

[Pen+22] Xue Bin Peng et al. "ASE: large-scale reusable adversarial skill embeddings for physically simulated characters". In: *ACM Trans. Graph.* 41.4 (July 2022). DOI: 10.1145/3528223.3530110.

[Pet14] Tibor Petzoldt. "On the relationship between pedestrian gap acceptance and time to arrival estimates". In: *Accident Analysis Prevention* 72 (2014), pp. 127–133. DOI: 10.1016/j.aap.2014.06.019. (Accessed: 7 Dec. 2024).

[Plu+05] Jodie M. Plumert et al. "Distance perception in real and virtual environments". In: *ACM Trans. Appl. Percept.* 2.3 (July 2005), pp. 216–233. DOI: 10.1145/1077399.1077402.

[PLY16] Eleonora Papadimitriou, Sylvain Lassarre, and George Yannis. "Introducing human factors in pedestrian crossing behaviour models". In: *Transportation Research Part F: Traffic Psychology and Behaviour* 36 (2016), pp. 69–82. DOI: 10.1016/j.trf.2015.11.003.

[PNB17] Tibor Petzoldt, Quyen Hoang-Sen Ngoc, and Katja Bogda. "Time to Arrival Estimates, (Pedestrian) Gap Acceptance and the Size Arrival Effect". In: *Driving Assessment Conference*. Vol. 9. Iowa City, Iowa: University of Iowa, Nov. 2017, pp. 44–50. URL: https://pubs.lib.uiowa.edu/driving/article/id/28421/ (Accessed: 7 Dec. 2024).

[PSM18] Atanas Poibrenski, Janis Sprenger, and Christian Müller. "Towards a methodology for training with synthetic data on the example of pedestrian detection in a frame-by-frame semantic segmentation task". In: *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*. SEFAIS '18. Gothenburg, Sweden: Association for Computing Machinery, 2018, pp. 31–34. DOI: 10.1145/3194085.3194093.

[PYG09] Eleonora Papadimitriou, George Yannis, and John Golias. "A critical assessment of pedestrian behaviour models". In: *Transportation Research Part F: Traffic Psychology and Behaviour* 12.3 (2009), pp. 242–255. DOI: 10.1016/j.trf.2008.12.004.

[PYG10] Eleonora Papadimitriou, George Yannis, and John Golias. "Theoretical Framework for Modeling Pedestrians' Crossing Behavior along a Trip". In: *Journal of Transportation Engineering* 136.10 (2010), pp. 914–924. DOI: 10.1061/(ASCE)TE.1943-5436.0000163.

[Qin+23] Zhongfei Qing et al. "Story-to-Motion: Synthesizing Infinite and Controllable Character Animation from Long Text". In: *SIGGRAPH Asia 2023 Technical Communications*. SA '23. Sydney, NSW, Australia: Association for Computing Machinery, 2023. DOI: 10.1145/3610543.3626176.

[Qui+09] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. 2009. URL: http://lars.mec.ua.pt/public/LAR%20Projects/BinPicking/2016_RodrigoSalgueiro/LIB/ROS/icraoss09-ROS.pdf (Accessed: 7 Dec. 2024).

[R C22] R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2022. URL: https://www.R-project.org/ (Accessed: 7 Dec. 2024).

[RA19] Alexandros Rouchitsas and Håkan Alm. "External Human–Machine Interfaces for Autonomous Vehicle-to-Pedestrian Communication: A Review of Empirical Work". In: *Frontiers in Psychology* 10 (2019). DOI: 10.3389/fpsyg.2019.02757.

[Rad+21] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 8748–8763. URL: https://proceedings.mlr.press/v139/radford21a.html (Accessed: 7 Dec. 2024).

[Ras+14] Björn Rasch et al. *Quantitative Methoden 2 Einführung in die Statistik für Psychologen und Sozialwissenschaftler*. Springer Berlin, 2014. URL: https://www.springer.com/de/book/9783642052705 (Accessed: 7 Dec. 2024).

[Red+23] Daniele Reda et al. "Physics-based Motion Retargeting from Sparse Inputs". In: *Proc. ACM Comput. Graph. Interact. Tech.* 6.3 (Aug. 2023). DOI: 10.1145/3606928.

[Ree+06] Reed, Matthew P. et al. "The HUMOSIM Ergonomics Framework: A New Approach to Digital Human Simulation for Ergonomic Analysis". In: *2006 Digital Human Modeling for Design and Engineering Conference*. SAE International, July 2006. DOI: 10.4271/2006-01-2365.

[Rem+21] Davis Rempe et al. "HuMoR: 3D Human Motion Model for Robust Pose Estimation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 11488–11499.

[Rem+23] Davis Rempe et al. "Trace and Pace: Controllable Pedestrian Animation via Guided Trajectory Diffusion". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 13756–13766. DOI: 10.1109/CVPR52729.2023.01322.

[Ric20] Thomas Richards. *A Review of Software for Crowd Simulation*. Tech. rep. 2020. URL: https://urban-analytics.github.io/dust/2020/03/PedSimReview.html (Accessed: 7 Dec. 2024).

[RKT17] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. *Agreeing to Cross: How Drivers and Pedestrians Communicate*. 2017. arXiv: 1702.03555.

[RKT18] Amir Rasouli, Iuliia Kotseruba, and John K. Tsotsos. "Understanding Pedestrian Behavior in Complex Traffic Scenes". In: *IEEE Transactions on Intelligent Vehicles* 3.1 (2018), pp. 61–70. DOI: 10.1109/TIV.2017.2788193.

[RN16] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.

[Rom+22] Robin Rombach et al. "High-Resolution Image Synthesis with Latent Diffusion Models". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 10674–10685. DOI: 10.1109/CVPR52688.2022.01042.

[Ros09] Tova Rosenbloom. "Crossing at a red light: Behaviour of individuals and groups". In: *Transportation Research Part F: Traffic Psychology and Behaviour* 12.5 (2009), pp. 389–394. DOI: 10.1016/j.trf.2009.05.002.

[SA03] V. P. Sisiopiku and D. Akin. "Pedestrian behaviors at and perceptions towards various pedestrian facilities: an examination based on observation and survey data". In: *Transportation Research Part F: Traffic Psychology and Behaviour* 6.4 (2003), pp. 249–274. DOI: 10.1016/j.trf.2003.06.001.

[Sal10] Brooke Salzman. "Gait and balance disorders in older adults". In: *American family physician* 82.1 (July 2010), pp. 61–68. ISSN: 0002-838X. URL: http://europepmc.org/abstract/MED/20590073 (Accessed: 7 Dec. 2024).

[San+13] Andrea Sanna et al. "Automatically Mapping Human Skeletons onto Virtual Character Armatures". In: *Intelligent Technologies for Interactive Entertainment*. Ed. by Matei Mancas et al. Cham: Springer International Publishing, 2013, pp. 80–89. ISBN: 978-3-319-03892-6.

[SB20] Sonja Schneider and Klaus Bengler. "Virtually the same? Analysing pedestrian behaviour by means of virtual reality". In: *Transportation Research Part F: Traffic Psychology and Behaviour* 68 (2020), pp. 231–256. DOI: 10.1016/j.trf.2019.11.005.

[Sch+12] David C. Schwebel et al. "Distraction and pedestrian safety: How talking on the phone, texting, and listening to music impact crossing the street". In: *Accident Analysis Prevention* 45 (2012), pp. 266–271. DOI: 10.1016/j.aap.2011.07.011.

[SF09] S. Schmidt and B. Färber. "Pedestrians at the kerb – Recognising the action intentions of humans". In: *Transportation Research Part F: Traffic Psychology and Behaviour* 12.4 (2009), pp. 300–310. DOI: 10.1016/j.trf.2009.02.003.

[SGS08] David C. Schwebel, Joanna Gaines, and Joan Severson. "Validation of virtual reality as a tool to understand and prevent child pedestrian injury". In: *Accident Analysis Prevention* 40.4 (2008), pp. 1394–1400. DOI: 10.1016/j.aap.2008.03.005.

[SH07] Alla Safonova and Jessica K. Hodgins. "Construction and optimal search of interpolated motion graphs". In: *ACM Trans. Graph.* 26.3 (July 2007), 106–es. DOI: 10.1145/1276377.1276510.

[Sha+18] Shital Shah et al. "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles". In: *Field and Service Robotics*. Ed. by Marco Hutter and Roland Siegwart. Cham: Springer International Publishing, 2018, pp. 621–635. ISBN: 978-3-319-67361-5.

[Shi+20] Soshi Shimada et al. "PhysCap: physically plausible monocular 3D motion capture in real time". In: *ACM Trans. Graph.* 39.6 (Nov. 2020). DOI: 10.1145/3414685.3417877.

[Shi+24] Yi Shi et al. *Interactive Character Control with Auto-Regressive Motion Diffusion Models*. 2024. arXiv: 2306.00416 [cs.CV].

[SHK22] Janis Sprenger, Lorena Hell, and Matthias Klusch. *Motion Capturing in Large-Scale VR Environments using Consumer Hardware*. Extended abstract presented at the 17th International Symposium of 3-D Analysis of Human Movement. Tokyo, July 2022.

[Sho+14] Alexander Shoulson et al. "ADAPT: The Agent Developmentand Prototyping Testbed". In: *IEEE Transactions on Visualization and Computer Graphics* 20.7 (2014), pp. 1035–1047. DOI: 10.1109/TVCG.2013.251.

[Shw21] Steve Shwartz. "Are Self-Driving Cars Really Safer Than Human Drivers?" In: *The Gradient* (2021). URL: https://thegradient.pub/are-self-driving-cars-really-safer-than-human-drivers (Accessed: 22 Sept. 2024).

[SJR03] Gordon Simpson, Lucy Johnston, and Michael Richardson. "An investigation of road crossing in a virtual environment". In: *Accident Analysis Prevention* 35.5 (2003), pp. 787–796. DOI: 10.1016/S0001-4575(02)00081-7.

[SMS15] Jiabin Shen, Leslie A. McClure, and David C. Schwebel. "Relations between temperamental fear and risky pedestrian behavior". In: *Accident Analysis Prevention* 80 (2015), pp. 178–184. DOI: 10.1016/j.aap.2015.04.011.

[Soc21] Society of Automobile Engineers. *Automated Driving - Levels of Driving Automation Are Defined in New Sae International Standard J3016_202104*. Tech. rep. 2021. URL: https://www.sae.org/standards/content/j3016_202104/ (Accessed: 7 Dec. 2024).

[Sof+21] Theodoros Sofianos et al. "Space-Time-Separable Graph Convolutional Network for Pose Forecasting". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 11209–11218.

[Soh+15] Jascha Sohl-Dickstein et al. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 2256–2265.

[Spr+19a] Janis Sprenger et al. "Capturing Subtle Motion Differences of Pedestrian Street Crossings". In: *Proceedings of the 32nd International Conference on Computer Animation and Social Agents*. CASA '19. Paris, France: Association for Computing Machinery, 2019, pp. 29–32. DOI: 10.1145/3328756.3328776.

[Spr+19b] Janis Sprenger et al. "Learning a Continuous Control of Motion Style from Natural Examples". In: *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games*. MIG '19. Newcastle upon Tyne, United Kingdom: Association for Computing Machinery, 2019. DOI: 10.1145/3359566.3360082.

[Spr+20] Janis Sprenger et al. "Variational Interpolating Neural Networks for Locomotion Synthesis". In: *Advances in Transdisciplinary Engineering* 11: DHM 2020.August (2020), pp. 71–81. DOI: 10.3233/ATDE200011.

[Spr+23] Janis Sprenger et al. "Cross-Cultural Behavior Analysis of Street-Crossing Pedestrians in Japan and Germany". In: *2023 IEEE Intelligent Vehicles Symposium (IV)*. 2023, pp. 1–7. DOI: 10.1109/IV55152.2023.10186635.

[Spr19] Janis Sprenger. "Pedestrian Behavior in Urban Environments - Influence of Pedestrian Variables on Behavior in a Multi-Agent Based Setting". Bachelor thesis. Saarland University, 2019.

[SPS09] David C. Schwebel, Danielle Dulion Pitts, and Despina Stavrinos. “The influence of carrying a backpack on college student pedestrian safety”. In: *Accident Analysis Prevention* 41.2 (2009), pp. 352–356. DOI: 10.1016/j.aap.2009.01.002.

[SSK09] David C. Schwebel, Despina Stavrinos, and Elizabeth M. Kongable. “Attentional control, high intensity pleasure, and risky pedestrian behavior in college students”. In: *Accident Analysis Prevention* 41.3 (2009), pp. 658–661. DOI: 10.1016/j.aap.2009.03.003.

[Sta+19] Sebastian Starke et al. “Neural state machine for character-scene interactions”. In: *ACM Trans. Graph.* 38.6 (Nov. 2019). DOI: 10.1145/3355089.3356505.

[Sta+20] Sebastian Starke et al. “Local motion phases for learning multi-contact character movements”. In: *ACM Trans. Graph.* 39.4 (Aug. 2020). DOI: 10.1145/3386569.3392450.

[Sta+21] Sebastian Starke et al. “Neural animation layering for synthesizing martial arts movements”. In: *ACM Trans. Graph.* 40.4 (July 2021). DOI: 10.1145/3450626.3459881.

[Sta+23] Paul Starke et al. “Motion In-Betweening with Phase Manifolds”. In: *Proc. ACM Comput. Graph. Interact. Tech.* 6.3 (Aug. 2023). DOI: 10.1145/3606921.

[Sta20] Sebastian Starke. “Bio IK: A memetic evolutionary algorithm for generic multi-objective inverse kinematics”. PhD thesis. Staats-und Universitätsbibliothek Hamburg Carl von Ossietzky, 2020. URL: https://ediss.sub.uni-hamburg.de/handle/ediss/8703 (Accessed: 7 Dec. 2024).

[Sta23] Statistisches Bundesamt (Destatis). *Height, weight and body mass index of the population by sex and age-groups*. 2023. URL: https://www.destatis.de/EN/Themes/Society-Environment/Health/Health-Status-Behaviour-Relevant-Health/Tables/liste-height-weight-body-mass-index-population-sex-age-groups.html (Accessed: 20 Sept. 2024).

[Suá21] Rolando Morales Suárez. “Natural and Accurate Stair Walking Synthesis for Character Control using Environment Information”. Master thesis. Saarland University, 2021.

[Sue+13] Cédric Sueur et al. “Different risk thresholds in pedestrian road crossing behaviour: A comparison of French and Japanese approaches”. In: *Accident Analysis Prevention* 58 (2013), pp. 59–63. DOI: 10.1016/j.aap.2013.04.027.

[Suj+18] Aaron Sujar et al. “Real-time animation of human characters’ anatomy”. In: *Computers Graphics* 74 (2018), pp. 268–277. DOI: 10.1016/j.cag.2018.05.025.

[Sun+22] Tao Sun et al. “SHIFT: A Synthetic Driving Dataset for Continuous Multi-Task Domain Adaptation”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 21339–21350. DOI: 10.1109/CVPR52688.2022.02068.

[SW65] S. S. Shapiro and M. B. Wilk. “An Analysis of Variance Test for Normality (Complete Samples)”. In: *Biometrika* 52.3/4 (1965), pp. 591–611. DOI: 10.2307/2333709. (Accessed: 21 July 2024).

[TDC16] Isabelle Tournier, Aurélie Dommes, and Viola Cavallo. “Review of safety and mobility issues among older pedestrians”. In: *Accident Analysis Prevention* 91 (2016), pp. 24–35. DOI: 10.1016/j.aap.2016.02.031.

[Tev+22] Guy Tevet et al. *Human Motion Diffusion Model*. 2022. arXiv: 2209.14916 [cs.CV].

[Tew+20] Ayush Tewari et al. *State of the Art on Neural Rendering*. 2020. arXiv: 2004.03805 [cs.CV].

[Tew+22] A. Tewari et al. “Advances in Neural Rendering”. In: *Computer Graphics Forum* 41.2 (2022), pp. 703–735. DOI: 10.1111/cgf.14507.

[TG11] Ariane Tom and Marie-Axelle Granié. “Gender differences in pedestrian rule compliance and visual search at signalized and unsignalized crossroads”. In: *Accident Analysis Prevention* 43.5 (2011), pp. 1794–1801. DOI: 10.1016/j.aap.2011.04.012.

[THB07] Lorenzo Torresani, Peggy Hackney, and Christoph Bregler. “Learning Motion Style Synthesis from Perceptual Observations”. In: *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*. The MIT Press, Sept. 2007. DOI: 10.7551/mitpress/7503.003.0179.

[Thi+08] Marcus Thiebaux et al. “SmartBody: behavior realization for embodied conversational agents”. In: *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*. AAMAS ’08. Estoril, Portugal: International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 151–158. ISBN: 9780981738109.

[THR11] Graham W. Taylor, Geoffrey E. Hinton, and Sam T. Roweis. “Two Distributed-State Models For Generating High-Dimensional Time Series”. In: *J. Mach. Learn. Res.* 12 (July 2011), pp. 1025–1068. ISSN: 1532-4435.

[Tou+23] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL].

[Tro02] Nikolaus F. Troje. “Decomposing biological motion: A framework for analysis and synthesis of human gait patterns”. In: *Journal of Vision* 2.5 (Sept. 2002), pp. 2–2. DOI: 10.1167/2.5.2.

[Tro13] Nikolaus F. Troje. “What Is Biological Motion? Definition, Stimuli, and Paradigms”. In: *Social Perception: Detection and Interpretation of Animacy, Agency, and Intention*. The MIT Press, Aug. 2013. DOI: 10.7551/mitpress/9780262019279.003.0002.

[TS10] N. F. Troje and S. Szabo. “Why is the average walker male?” In: *Journal of Vision* 6.6 (2010). DOI: 10.1167/6.6.1034.

[Tyr+09] Richard A. Tyrrell et al. “Seeing pedestrians at night: Visual clutter does not mask biological motion”. In: *Accident Analysis Prevention* 41.3 (2009), pp. 506–512. DOI: 10.1016/j.aap.2009.02.001.

[VFR] VFR Verlag für Rechtsjournalismus GmbH. *Bußgeldkatalog für Fußgänger 2021 - Regeln im Straßenverkehr*. URL: https://www.bussgeldkatalog.org/fussgaenger/ (Accessed: 21 July 2024).

[Vil+18] Ruben Villegas et al. “Neural Kinematic Networks for Unsupervised Motion Retargetting”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8639–8648. DOI: 10.1109/CVPR.2018.00901.

[Vil+21] Ruben Villegas et al. “Contact-Aware Retargeting of Skinned Motion”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 9700–9709. DOI: 10.1109/ICCV48922.2021.00958.

[Vin81] Marja P. Vinje. “Children as pedestrians: Abilities and limitations”. In: *Accident Analysis Prevention* 13.3 (1981). Special Issue: Traffic education for young pedestrians, pp. 225–240. DOI: 10.1016/0001-4575(81)90006-3.

[Voz+23] Igor Vozniak et al. “Context-Empowered Visual Attention Prediction in Pedestrian Scenarios”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2023, pp. 950–960.

[Wan+22] Huarong Wang et al. “The effect of age and sensation seeking on pedestrian crossing safety in a virtual reality street”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 88 (2022), pp. 99–110. DOI: 10.1016/j.trf.2022.05.010.

[Wan+23] Jiashun Wang et al. “Zero-shot Pose Transfer for Unrigged Stylized 3D Characters”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 8704–8714. DOI: 10.1109/CVPR52729.2023.00841.

[Wan+24] Jiashun Wang et al. “Multi-person 3D motion prediction with multi-range transformers”. In: *Proceedings of the 35th International Conference on Neural Information Processing Systems*. NIPS ’21. Red Hook, NY, USA: Curran Associates Inc., 2024. ISBN: 9781713845393.

[WCW14] Xin Wang, Qiudi Chen, and Wanliang Wang. “3D Human Motion Editing and Synthesis: A Survey”. In: *Computational and Mathematical Methods in Medicine* 2014.11 (2014), pp. 1–11. DOI: 10.1155/2014/104535.

[WCX21] Zhiyong Wang, Jinxiang Chai, and Shihong Xia. “Combining Recurrent Neural Networks and Adversarial Training for Human Motion Synthesis and Control”. In: *IEEE Transactions on Visualization and Computer Graphics* 27.1 (2021), pp. 14–28. DOI: 10.1109/TVCG.2019.2938520.

[Wei99] Gerhard Weiss, ed. *Multiagent systems: a modern approach to distributed artificial intelligence*. Cambridge, MA, USA: MIT Press, 1999. ISBN: 0262232030.

[WFF19] Axel Wolfermann, Bernhard Friedrich, and Martin Fellendorf. “4 - Germany and Austria”. In: *Global Practices on Road Traffic Signal Control*. Ed. by Keshuang Tang et al. Elsevier, 2019, pp. 37–67. DOI: 10.1016/B978-0-12-815302-4.00005-4.

[WFH08] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. “Gaussian Process Dynamical Models for Human Motion”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.2 (2008), pp. 283–298. DOI: 10.1109/TPAMI.2007.1167.

[WH20] Yuxin Wu and Kaiming He. “Group Normalization”. In: *International Journal of Computer Vision* 128.3 (Mar. 2020), pp. 742–755. DOI: 10.1007/s11263-019-01198-w.

[Wu+13] Chenglei Wu et al. “On-set performance capture of multiple actors with a stereo camera”. In: *ACM Trans. Graph.* 32.6 (Nov. 2013). DOI: 10.1145/2508363.2508418.

[WZC12] Xiaolin Wei, Peizhao Zhang, and Jinxiang Chai. “Accurate realtime full-body motion capture using a single depth camera”. In: vol. 31. 6. New York, NY, USA: Association for Computing Machinery, Nov. 2012. DOI: 10.1145/2366145.2366207.

[Xia+15] Shihong Xia et al. “Realtime style transfer for unlabeled heterogeneous human motion”. In: *ACM Trans. Graph.* 34.4 (July 2015). DOI: 10.1145/2766999.

[Xia+17] Shihong Xia et al. “A Survey on Human Performance Capture and Animation”. In: *Journal of Computer Science and Technology* 32.3 (May 2017), pp. 536–554. DOI: 10.1007/s11390-017-1742-y.

[Xia10] Shihong Xia. “Modeling style and variation in human motion”. In: *2010 4th International Universal Communication Symposium*. 2010, pp. 207–207. DOI: 10.1109/IUCS.2010.5666642.

[Yam01] Ikushi Yamaguchi. “Perceived organizational support for satisfying autonomy needs of Japanese white-collar workers: A comparison between Japanese and US-affiliated companies”. In: *Journal of Managerial Psychology* 16.6 (2001), pp. 434–448. DOI: 10.1108/EUM0000000005773.

[Yan+20] Shanwen Yang et al. “A review on crowd simulation and modeling”. In: *Graphical Models* 111 (2020), p. 101081. DOI: 10.1016/j.gmod.2020.101081.

[Yan+23] Yuhang Yang et al. “Suicidal Pedestrian: Generation of Safety-Critical Scenarios for Autonomous Vehicles”. In: *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. 2023, pp. 1983–1988. DOI: 10.1109/ITSC57777.2023.10422034.

[YM16] M. Ersin Yumer and Niloy J. Mitra. “Spectral style transfer for human motion between independent actions”. In: vol. 35. 4. New York, NY, USA: Association for Computing Machinery, July 2016. DOI: 10.1145/2897824.2925955.

[Yu+20] Fisher Yu et al. “BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2633–2642. DOI: 10.1109/CVPR42600.2020.00271.

[Yua+23] Ye Yuan et al. “PhysDiff: Physics-Guided Human Motion Diffusion Model”. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 15964–15975. DOI: 10.1109/ICCV51070.2023.01467.

[ZB23] Chi Zhang and Christian Berger. “Pedestrian Behavior Prediction Using Deep Learning Methods for Urban Scenarios: A Review”. In: *IEEE Transactions on Intelligent Transportation Systems* 24.10 (2023), pp. 10279–10301. DOI: 10.1109/TITS.2023.3281393.

[ZBS17] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. “CityPersons: A Diverse Dataset for Pedestrian Detection”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4457–4465. DOI: 10.1109/CVPR.2017.474.

[Zha+18] He Zhang et al. “Mode-adaptive neural networks for quadruped motion control”. In: *ACM Trans. Graph.* 37.4 (July 2018). DOI: 10.1145/3197517.3201366.

[Zha+23a] Jianrong Zhang et al. “Generating Human Motion from Textual Descriptions with Discrete Representations”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 14730–14740. DOI: 10.1109/CVPR52729.2023.01415.

[Zha+23b] Mingyuan Zhang et al. “ReMoDiffuse: Retrieval-Augmented Motion Diffusion Model”. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 364–373. DOI: 10.1109/ICCV51070.2023.00040.

[Zha+24a] Chi Zhang et al. “Predicting and Analyzing Pedestrian Crossing Behavior at Unsignalized Crossings”. In: *2024 IEEE Intelligent Vehicles Symposium (IV)*. 2024, pp. 674–681. DOI: 10.1109/IV55156.2024.10588752.

[Zha+24b] Chi Zhang et al. “Predicting Pedestrian Crossing Behavior in Germany and Japan: Insights into Model Transferability”. In: *IEEE Transactions on Intelligent Vehicles* (2024), pp. 1–16. DOI: 10.1109/TIV.2024.3506727.

[Zha+24c] Mingyuan Zhang et al. *MotionDiffuse: Text-Driven Human Motion Generation With Diffusion Model*. 2024. DOI: 10.1109/TPAMI.2024.3355414.

[Zho+14] Liuyang Zhou et al. “Human motion variation synthesis with multivariate Gaussian processes”. In: *Computer Animation and Virtual Worlds* 25.3-4 (2014), pp. 301–309. DOI: 10.1002/cav.1599.

[Zho+18] Yi Zhou et al. *Auto-Conditioned Recurrent Networks for Extended Complex Human Motion Synthesis*. 2018. arXiv: 1707.05363.

[ZK03] M. Suzanne Zeedyk and Laura Kelly. "Behavioural observations of adult–child pairs at pedestrian crossings". In: *Accident Analysis Prevention* 35.5 (2003), pp. 771–776. DOI: 10.1016/S0001-4575(02)00086-6.

[ZNW19] Xin Zhang, Hideki Nakamura, and Yu Wu. "Analyzing the Impact of Refuge Islands on Pedestrian and Driver Behavior at Unsignalized Mid-block Crosswalks". In: *World Conference on Transport Research*. Mumbai, India: Transportation Research Procedia, 2019, pp. 1–8. URL: https://www.wctrs-society.com/wp-content/uploads/abstracts/mumbai/WCTR2019_Mumbai_1206.pdf (Accessed: 7 Dec. 2024).

[ZRA20] Niaz Mahmud Zafri, Atikul Islam Rony, and Neelopal Adri. "Study on Pedestrian Compliance Behavior at Vehicular Traffic Signals and Traffic-Police-Controlled Intersections". In: *International Journal of Intelligent Transportation Systems Research* 18.3 (Sept. 2020), pp. 400–411. DOI: 10.1007/s13177-019-00208-y.

[ZW11] Xiangling Zhuang and Changxu Wu. "Pedestrians' crossing behaviors and safety at unmarked roadway in China". In: *Accident Analysis Prevention* 43.6 (2011), pp. 1927–1936. DOI: 10.1016/j.aap.2011.05.005.

List of Figures

List of Tables

Appendix A
Appendix

A.1 Additional Mathematical Functions

A.1.1 ELU activation function

The exponential linear units (ELU) can be expressed as:

$$ELU(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha * (exp(x) - 1), & \text{if } x \leq 0 \end{cases} \tag{A.1}$$

A.1.2 Kronecker Product

The Kronecker product is a specialization of a tensor product combining a matrix A of size $m \times n$ and a matrix B of size $p \times q$ to a matrix of size $pm \times qn$:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix} \tag{A.2}$$

A.1.3 Cubic Catmull-Rom Spline

A single cubic Catmull-Rom spline with four control points $\beta = \{\alpha_0\ \alpha_1\ \alpha_2\ \alpha_3\}$ can be defined as

$$\begin{aligned}\phi(p;\beta) &= \alpha_{k_1} \\ &\quad + w(\frac{1}{2}\alpha_{k_2} - \frac{1}{2}\alpha_{k_0}) \\ &\quad + w^2(\alpha_{k_0} - \frac{5}{2}\alpha_{k_1} + 2\alpha_{k_2} - \frac{1}{2}\alpha_{k_3}) \\ &\quad + w^3(\frac{3}{2}\alpha_{k_1} - \frac{3}{2}\alpha_{k_2} + \frac{1}{2}\alpha_{k_3} - \frac{1}{2}\alpha_{k_0}) \\ w &= \frac{4p}{2\pi} \mod 1 \\ k_n &= (\left\lfloor \frac{4p}{2\pi} \right\rfloor + n - 1) \mod 4\end{aligned} \tag{A.3}$$

A.2 MOSIM Definition

A.2.1 Intermediate Skeleton Description

Listing A.1: Implementation of the intermediate skeleton structure.

```
public static MQuaternion identityQ = new MQuaternion(0, 0, 0, 1);
public static List<MChannel> defaultJointChannels = new List<MChannel>() {
    MChannel.WRotation, MChannel.XRotation, MChannel.YRotation, MChannel.ZRotation};
public static List<MChannel> defaultRootChannels = new List<MChannel>() {
    MChannel.XOffset, MChannel.YOffset, MChannel.ZOffset,
    MChannel.WRotation, MChannel.XRotation, MChannel.YRotation, MChannel.ZRotation };
public static List<MChannel> zeroChannels = new List<MChannel>();
private static MQuaternion qRotY180 = new MQuaternion(0, 1, 0, 0);
private static MQuaternion qRotYM180 = new MQuaternion(0, -1, 0, 0);

private static MQuaternion qRotY90 = new MQuaternion(0, 0.707106769, 0, 0.707106769);
private static MQuaternion qRotYM90 = new MQuaternion(0, -0.707106769, 0, 0.707106769);

private static MQuaternion qRotX90 = new MQuaternion(0.707106769, 0, 0, 0.707106769);
private static MQuaternion qRotXM90 = new MQuaternion(-0.707106769, 0, 0, 0.707106769);
private static MQuaternion qRotX180 = new MQuaternion(1, 0, 0, 0);
private static MQuaternion qRotXM180 = new MQuaternion(-1, 0, 0, 0);

private static MQuaternion qRotZ90 = new MQuaternion(0, 0, 0.707106769, 0.707106769);
private static MQuaternion qRotZM90 = new MQuaternion(0, 0, -0.707106769, 0.707106769);
private static MQuaternion qRotZ180 = new MQuaternion(0, 0, 1, 0);
private static MQuaternion qRotZM180 = new MQuaternion(0, 0, -1, 0);


private static MJoint NewMJoint(string id, MJointType type, MVector3 offset, MQuaternion
    rotation, string parentID, List<MChannel>channels)
{
```

```
    MJoint j = new MJoint(id, type, offset, rotation);
    j.Parent = parentID;
    j.Channels = channels;
    return j;
}
public static List<MJoint> GetDefaultJointList()
{
    List<MJoint> defaultJoints = new List<MJoint>() {
    // 7 joints along the spine (6 animated, 39 channels)
    NewMJoint("Root", MJointType.Root, new MVector3(0,0,0), identityQ, null,
        defaultRootChannels),
    NewMJoint("PelvisCenter", MJointType.PelvisCentre, new MVector3(0, 0, 0), qRotY90,
        "Root", defaultRootChannels),
    NewMJoint("S1L5Joint", MJointType.S1L5Joint, new MVector3(0, 0.18, 0), identityQ,
        "PelvisCenter", defaultRootChannels),
    NewMJoint("T12L1Joint", MJointType.T12L1Joint, new MVector3(0, 0.15, 0), identityQ,
        "S1L5Joint", defaultRootChannels), // 0.33 - 0.18
    NewMJoint("T1T2Joint", MJointType.T1T2Joint, new MVector3(0, 0.43, 0), identityQ,
        "T12L1Joint", defaultRootChannels), // 0.76 - 0.33
    NewMJoint("C4C5Joint", MJointType.C4C5Joint, new MVector3(0, 0.11, 0), identityQ,
        "T1T2Joint", defaultRootChannels), // 0.87 - 0.76
    NewMJoint("HeadJoint", MJointType.HeadJoint, new MVector3(0, 0.13, 0), qRotYM180,
        "C4C5Joint", defaultJointChannels), // 1.0 - 0.87
    NewMJoint("HeadTip", MJointType.HeadTip, new MVector3(0, 0.16, 0), identityQ,
        "HeadJoint", zeroChannels),


    // Left Arm: 3 joints (3 animated, 15 channels)
    NewMJoint("LeftShoulder", MJointType.LeftShoulder, new MVector3(0, 0, -1),
        qRotXM90.Multiply(qRotYM90), "T1T2Joint", defaultRootChannels),
    NewMJoint("LeftElbow", MJointType.LeftElbow, new MVector3(0, 1, 0), qRotY90,
        "LeftShoulder", defaultJointChannels),
    NewMJoint("LeftWrist", MJointType.LeftWrist, new MVector3(0, 1, 0), qRotY90,
        "LeftElbow", defaultJointChannels),

    //left hand: 20 joints (16 animated, 75 channels)
    NewMJoint("LeftMiddleProximal", MJointType.LeftMiddleProximal, new MVector3(0, 1, 0),
        identityQ, "LeftWrist", defaultRootChannels),
    NewMJoint("LeftMiddleMeta", MJointType.LeftMiddleMeta, new MVector3(0, 1, 0),
        identityQ, "LeftMiddleProximal", defaultJointChannels),
    NewMJoint("LeftMiddleDistal", MJointType.LeftMiddleDistal, new MVector3(0, 1, 0),
        identityQ, "LeftMiddleMeta", defaultJointChannels),
    NewMJoint("LeftMiddleTip", MJointType.LeftMiddleTip, new MVector3(0, 0.03, 0),
        identityQ, "LeftMiddleDistal", zeroChannels),

    NewMJoint("LeftIndexProximal", MJointType.LeftIndexProximal, new MVector3(0, 1, 0),
        identityQ, "LeftWrist", defaultRootChannels),
    NewMJoint("LeftIndexMeta", MJointType.LeftIndexMeta, new MVector3(0, 1, 0), identityQ,
        "LeftIndexProximal", defaultJointChannels),
    NewMJoint("LeftIndexDistal", MJointType.LeftIndexDistal, new MVector3(0, 1, 0),
```

```
    identityQ, "LeftIndexMeta", defaultJointChannels),
NewMJoint("LeftIndexTip", MJointType.LeftIndexTip, new MVector3(0, 0.03, 0),
    identityQ, "LeftIndexDistal", zeroChannels),

NewMJoint("LeftRingProximal", MJointType.LeftRingProximal, new MVector3(0, 1, 0),
    identityQ, "LeftWrist", defaultRootChannels),
NewMJoint("LeftRingMeta", MJointType.LeftRingMeta, new MVector3(0, 1, 0), identityQ,
    "LeftRingProximal", defaultJointChannels),
NewMJoint("LeftRingDistal", MJointType.LeftRingDistal, new MVector3(0, 1, 0),
    identityQ, "LeftRingMeta", defaultJointChannels),
NewMJoint("LeftRingTip", MJointType.LeftRingTip, new MVector3(0, 0.03, 0), identityQ,
    "LeftRingDistal", zeroChannels),

NewMJoint("LeftLittleProximal", MJointType.LeftLittleProximal, new MVector3(0, 1, 0),
    identityQ, "LeftWrist", defaultRootChannels),
NewMJoint("LeftLittleMeta", MJointType.LeftLittleMeta, new MVector3(0, 1, 0),
    identityQ, "LeftLittleProximal", defaultJointChannels),
NewMJoint("LeftLittleDistal", MJointType.LeftLittleDistal, new MVector3(0, 1, 0),
    identityQ, "LeftLittleMeta", defaultJointChannels),
NewMJoint("LeftLittleTip", MJointType.LeftLittleTip, new MVector3(0, 0.03, 0),
    identityQ, "LeftLittleDistal", zeroChannels),

NewMJoint("LeftThumbMid", MJointType.LeftThumbMid, new MVector3(0, 1, 0),
    new MQuaternion(-0.3826834559440613, 0, 0,  0.9238795042037964), "LeftWrist",
        defaultRootChannels),
NewMJoint("LeftThumbMeta", MJointType.LeftThumbMeta, new MVector3(0, 1, 0), identityQ,
    "LeftThumbMid", defaultJointChannels),
NewMJoint("LeftThumbCarpal", MJointType.LeftThumbCarpal, new MVector3(0, 1, 0),
    identityQ, "LeftThumbMeta", defaultJointChannels),
NewMJoint("LeftThumbTip", MJointType.LeftThumbTip, new MVector3(0, 0.03, 0),
    identityQ, "LeftThumbCarpal", zeroChannels),


// Right Arm: 3 joints (3 animated, 15 channels)
NewMJoint("RightShoulder", MJointType.RightShoulder, new MVector3(0, 0, 1),
    qRotX90.Multiply(qRotY90), "T1T2Joint", defaultRootChannels),
NewMJoint("RightElbow", MJointType.RightElbow, new MVector3(0, 1, 0), qRotYM90,
    "RightShoulder", defaultJointChannels),
NewMJoint("RightWrist", MJointType.RightWrist, new MVector3(0, 1, 0), qRotYM90,
    "RightElbow", defaultJointChannels),

//left hand: 20 joints (16 animated, 75 channels)
NewMJoint("RightMiddleProximal", MJointType.RightMiddleProximal,
    new MVector3(0, 1, 0), identityQ, "RightWrist", defaultRootChannels),
NewMJoint("RightMiddleMeta", MJointType.RightMiddleMeta, new MVector3(0, 1, 0),
    identityQ, "RightMiddleProximal", defaultJointChannels),
NewMJoint("RightMiddleDistal", MJointType.RightMiddleDistal, new MVector3(0, 1, 0),
    identityQ, "RightMiddleMeta", defaultJointChannels),
NewMJoint("RightMiddleTip", MJointType.RightMiddleTip, new MVector3(0, 0.03, 0),
    identityQ, "RightMiddleDistal", zeroChannels),
```

```
NewMJoint("RightIndexProximal", MJointType.RightIndexProximal, new MVector3(0, 1, 0),
    identityQ, "RightWrist", defaultRootChannels),
NewMJoint("RightIndexMeta", MJointType.RightIndexMeta, new MVector3(0, 1, 0),
    identityQ, "RightIndexProximal", defaultJointChannels),
NewMJoint("RightIndexDistal", MJointType.RightIndexDistal, new MVector3(0, 1, 0),
    identityQ, "RightIndexMeta", defaultJointChannels),
NewMJoint("RightIndexTip", MJointType.RightIndexTip, new MVector3(0, 0.03, 0),
    identityQ, "RightIndexDistal", zeroChannels),

NewMJoint("RightRingProximal", MJointType.RightRingProximal, new MVector3(0, 1, 0),
    identityQ, "RightWrist", defaultRootChannels),
NewMJoint("RightRingMeta", MJointType.RightRingMeta, new MVector3(0, 1, 0), identityQ,
    "RightRingProximal", defaultJointChannels),
NewMJoint("RightRingDistal", MJointType.RightRingDistal, new MVector3(0, 1, 0),
    identityQ, "RightRingMeta", defaultJointChannels),
NewMJoint("RightRingTip", MJointType.RightRingTip, new MVector3(0, 0.03, 0),
    identityQ, "RightRingDistal", zeroChannels),

NewMJoint("RightLittleProximal", MJointType.RightLittleProximal,
    new MVector3(0, 1, 0), identityQ, "RightWrist", defaultRootChannels),
NewMJoint("RightLittleMeta", MJointType.RightLittleMeta, new MVector3(0, 1, 0),
    identityQ, "RightLittleProximal", defaultJointChannels),
NewMJoint("RightLittleDistal", MJointType.RightLittleDistal, new MVector3(0, 1, 0),
identityQ, "RightLittleMeta", defaultJointChannels),
NewMJoint("RightLittleTip", MJointType.RightLittleTip, new MVector3(0, 0.03, 0),
    identityQ, "RightLittleDistal", zeroChannels),

NewMJoint("RightThumbMid", MJointType.RightThumbMid, new MVector3(0, 1, 0),
    new MQuaternion(0.3826834559440613, 0, 0,  0.9238795042037964), "RightWrist",
    defaultRootChannels),//0.53730, 0, 0, 0.84339
NewMJoint("RightThumbMeta", MJointType.RightThumbMeta, new MVector3(0, 1, 0),
    identityQ, "RightThumbMid", defaultJointChannels),
NewMJoint("RightThumbCarpal", MJointType.RightThumbCarpal, new MVector3(0, 1, 0),
    identityQ, "RightThumbMeta", defaultJointChannels),
NewMJoint("RightThumbTip", MJointType.RightThumbTip, new MVector3(0, 0.03, 0),
    identityQ, "RightThumbCarpal", zeroChannels),

// Left leg: 5 joints (4 animated, 16 channels)
NewMJoint("LeftHip", MJointType.LeftHip, new MVector3(0, 0, -1), qRotZ180,
    "PelvisCenter", defaultJointChannels),
NewMJoint("LeftKnee", MJointType.LeftKnee, new MVector3(0, 1, 0), identityQ,
    "LeftHip", defaultJointChannels),
NewMJoint("LeftAnkle", MJointType.LeftAnkle, new MVector3(0, 1, 0),
    new MQuaternion(0, 0, -0.53730, 0.84339), "LeftKnee", defaultJointChannels),
NewMJoint("LeftBall", MJointType.LeftBall, new MVector3(0, 1, 0),
    new MQuaternion(0, 0, -0.2164396196603775,  0.9762960076332092), "LeftAnkle",
    defaultJointChannels),
NewMJoint("LeftBallTip", MJointType.LeftBallTip, new MVector3(0, 0.08, 0), identityQ,
    "LeftBall", zeroChannels),
```

```
    // Right leg: 5 joints (4 animated, 16 channels)
    NewMJoint("RightHip", MJointType.RightHip, new MVector3(0, 0, 1), qRotZ180,
        "PelvisCenter", defaultJointChannels),
    NewMJoint("RightKnee", MJointType.RightKnee, new MVector3(0, 1, 0), identityQ,
        "RightHip", defaultJointChannels),
    NewMJoint("RightAnkle", MJointType.RightAnkle, new MVector3(0, 1, 0),
        new MQuaternion(0, 0, -0.53730, 0.84339), "RightKnee", defaultJointChannels),
    NewMJoint("RightBall", MJointType.RightBall, new MVector3(0, 1, 0),
        new MQuaternion(0, 0, -0.2164396196603775, 0.9762960076332092), "RightAnkle",
        defaultJointChannels),
    NewMJoint("RightBallTip", MJointType.RightBallTip, new MVector3(0, 0.08, 0),
        identityQ, "RightBall", zeroChannels)
    };
    return defaultJoints;
}
```

A.3 Intermediate Skeleton

A.3.1 Proof of Equality under Subsequent Retargeting

The joint rotation and position in the reference pose are defined as (R_i, P_i) for the intermediate (R_t, P_t) and the target skeleton. The position and rotation of a joint in a specific pose are defined as (r_i, p_i) for the intermediate and (r_t, p_t) for the target skeleton.

Equality is assumed when retargeting a joint rotation from the intermediate to the target and back to the intermediate skeleton. It can be shown by inserting the respective retargeting functions:

$$
\begin{aligned}
r_i &= r_t * R_t^{-1} * R_i \\
r_i &= r_i * R_i^{-1} * R_t * R_t^{-1} * R_i \\
r_i &= r_i \\
0 &= 0
\end{aligned}
$$

The same can be shown for retargeting a joint rotation from the target to the intermediate and back to the target skeleton. The proof is trivial.

For a joint position, the same equality is assumed. It can be shown by inserting the respective functions repeatedly:

$$
\begin{aligned}
p_i &= p_t + r_t * O_t \\
p_i &= p_i + r_i * O_i + r_t * O_t \\
-(r_i * O_i) &= r_t * O_t \\
-(r_i * R_i^{-1} * (P_t - P_i)) &= r_t * O_t \\
-(r_i * R_i^{-1} * (P_t - P_i)) &= r_i * R_i^{-1} * R_t * O_t \\
-(P_t - P_i) &= R_t * O_t \\
-(P_t - P_i) &= R_t * R_t^{-1} * (P_i - P_t) \\
-(P_t - P_i) &= P_i - P_t \\
P_i - P_t &= P_i - P_t \\
0 &= 0
\end{aligned}
$$

The same can be shown for retargeting a joint position from the target to the intermediate and back to the target skeleton. The proof is trivial.

Thus, mathematical equality under subsequent retargeting steps is proven.

A.4 Software and Hardware

A.4.1 Software

The following software is either used or referenced in this dissertation.

AnyLogic Simulation Software	AnyLogic North America, LLC `https://www.anylogic.de/`
Apache Thrift	Apache Software Foundation `https://thrift.apache.org/`
Blender	Blender Foundation `https://www.blender.org/`
Car Maker	IPG Automotive GmbH `https://www.ipg-automotive.com/de/produkte-loesungen/software/carmaker/`
Datasmith	Epic Games, Inc. `https://www.unrealengine.com/en-US/datasmith`
DAZ 3D	DAZ Productions, Inc. `https://www.daz3d.com/`

Filmbox file format (FBX)	Autodesk Inc. https://www.autodesk.com/products/fbx/overview
FinalIK	RootMotion http://root-motion.com/
Godot	Godot Foundation https://godotengine.org/
IBM SPSS Statistics	IBM Deutschland GmbH https://www.ibm.com/de-de/products/spss-statistics
iClone	Reallusion, Inc. https://www.reallusion.com/de/iclone/
IPS Imma	Industrial Path Solutions Sweden AB https://industrialpathsolutions.com/ips-imma
MakeHuman	The MakeHuman Community http://www.makehumancommunity.org/
MassMotion	Arup Group Limited https://www.arup.com/services/digital-solutions-and-tools/massmotion/
Maya	Autodesk, Inc. https://www.autodesk.com/de/products/maya
Mecanim	Unity Technologies https://docs.unity3d.com/Manual/AnimationOverview.html
Metahuman	Epic Games, Inc. https://www.unrealengine.com/en-US/metahuman
Mixamo	Adobe Systems Incorporated https://www.mixamo.com/
MotionBuilder	Autodesk, Inc. https://www.autodesk.com/products/motionbuilder
NVIDIA Omniverse	Nvidia Corporation https://www.nvidia.com/de-de/omniverse/

NVIDIA DriveSim	Nvidia Corporation `https://developer.nvidia.com/drive`
Panda3D	Carnegie Mellon University `https://www.panda3d.org/`
Pixyz	Unity Technologies `https://unity.com/products/pixyz`
Poser	Bondware, Inc. `https://www.posersoftware.com/`
RAMSIS	Humanetics Digital Europe GmbH `https://www.human-solutions.com/de/produkte/ramsis-allgemein/index.html`
Ready Player Me	Ready Player Me `https://readyplayer.me/de`
realvirtual.io	realvirtual GmbH `https://realvirtual.io/`
ReplicaR	Automotive Artificial Intelligence (AAI) GmbH `https://www.automotive-ai.com/`
Sora	OpenAI `https://openai.com/index/sora/`
Spore	Electronic Arts, Inc. `https://www.spore.com/`
STEPS	Mott MacDonald Group Limited `https://www.steps.mottmac.com/steps-dynamics`
TRONIS	TWT GmbH `https://twt-innovation.de/produkte/`
Unity	Unity Technologies `https://unity.com/`
Unreal Engine	Epic Games, Inc. `https://www.unrealengine.com`

A.4.2 Hardware

The following hardware is either used or referenced in this work.

OptiTrack	NaturalPoint, Inc. https://www.optitrack.com/
Hololens 2	Microsoft Cooperation https://learn.microsoft.com/de-de/hololens/hololens2-hardware
Perception Neuron	Noitom Ltd. https://www.noitom.com/perception-neuron-series
The Captury	DARI Motion, Inc https://captury.com/
Vicon	Vicon Motion Systems Ltd https://www.vicon.com/
Vive XR Elite	HTC Cooperation https://www.vive.com/de/product/vive-xr-elite
Vive Pro Eye	HTC Cooperation https://www.vive.com/sea/product/vive-pro-eye
XSens MVN Awinda	Movella, Inc. https://www.movella.com/products/motion-capture/xsens-mvn-awinda