

Entwicklung eines ereignisgesteuerten Animationssystems für performative Bewegungsabläufe am Beispiel autonomer Torjubiläum im Roboterfußball

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)
im Studiengang Digitale Medien

Eingereicht von:

Aaron Postels
Matrikelnr.: 6277067
postels@uni-bremen.de

Betreuer / Gutachter:

1. Prof. Dr.-Ing. Udo Frese
2. Nuri Ovüc

Bremen, den 23.07.2026

Erklärungen

A) Eigenständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Teile meiner Arbeit, die wortwörtlich oder dem Sinn nach anderen Werken entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht. Gleiches gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet, dazu zählen auch KI-basierte Anwendungen oder Werkzeuge. Die Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht. Die elektronische Fassung der Arbeit stimmt mit der gedruckten Version überein. Mir ist bewusst, dass wahrheitswidrige Angaben als Täuschung behandelt werden.

- ☒ Ich habe KI-basierte Anwendungen und/oder Werkzeuge genutzt und diese im Anhang „Nutzung KI-basierte Anwendungen“ dokumentiert.
- ☐ Ich habe keine KI-basierten Anwendungen oder Werkzeuge genutzt.

B) Erklärung zur Veröffentlichung von Bachelor- oder Masterarbeiten

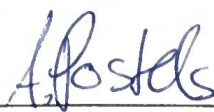
Die Abschlussarbeit wird zwei Jahre nach Studienabschluss dem Archiv der Universität Bremen zur dauerhaften Archivierung angeboten. Archiviert werden 1. Masterarbeiten mit lokalem oder regionalem Bezug sowie pro Studienfach und Studienjahr 10% aller Abschlussarbeiten und 2. Bachelorarbeiten des jeweils ersten und letzten Bachelorabschlusses pro Studienfach und Jahr.

- ☒ Ich bin damit einverstanden, dass meine Abschlussarbeit im Universitätsarchiv für wissenschaftliche Zwecke von Dritten eingesehen werden darf.
- ☐ Ich bin damit einverstanden, dass meine Abschlussarbeit nach _____ Jahren gem. § 5 Abs. 6 BremArchivG im Universitätsarchiv für wissenschaftliche Zwecke von Dritten eingesehen werden darf.
- ☐ Ich bin nicht damit einverstanden, dass meine Abschlussarbeit im Universitätsarchiv für wissenschaftliche Zwecke von Dritten eingesehen werden darf.

C) Einverständniserklärung zur Überprüfung der elektronischen Fassung durch Plagiatsoftware

Eingereichte Arbeiten können nach § 18 des Allgemeinen Teil der Bachelor- bzw. der Masterprüfungsordnungen der Universität Bremen mit qualifizierter Software auf Plagiatsvorwürfe untersucht werden. Zum Zweck der Überprüfung auf Plagiate erfolgt das Hochladen auf den Server der von der Universität Bremen aktuell genutzten Plagiatsoftware.

- ☒ Ich bin damit einverstanden, dass die von mir vorgelegte und verfasste Arbeit zum oben genannten Zweck dauerhaft auf dem externen Server der aktuell von der Universität Bremen genutzten Plagiatsoftware, in einer institutionseigenen Bibliothek (Zugriff nur durch die Universität Bremen), gespeichert wird.
- ☐ Ich bin nicht damit einverstanden, dass die von mir vorgelegte und verfasste Arbeit zum o.g. Zweck dauerhaft auf dem externen Server der aktuell von der Universität Bremen genutzten Plagiatsoftware gespeichert wird.



Bremen, den 21.07.2026

Aaron Postels

Zusammenfassung

Mit dem Wechsel vom kleineren NAO-Roboter zum größeren Booster K1 verändert sich das Erscheinungsbild des Roboterfußballs. Der K1 wirkt durch seine eher technische Gestaltung deutlich nüchterner als sein Vorgänger. Diese Arbeit greift die Idee auf, dem Roboter durch sichtbare Reaktionen und einen Torjubel mehr Persönlichkeit zu geben.

Dafür wurde eine durchgängige Pipeline entwickelt, mit der Tanzbewegungen erstellt, bearbeitet und in das B-Human-System eingebunden werden können. Als Beispiel begleitet der Macarena-Tanz die einzelnen Schritte der Arbeit. Eine Videoaufnahme wird zunächst in eine menschliche 3D-Bewegung und danach in eine Bewegung des K1 übertragen. Das im Rahmen der Arbeit entwickelte Blender-Add-on "K1 Animator" macht diese Daten anschließend direkt am Robotermodell sichtbar und bearbeitbar. Bewegungen können dort auch vollständig von Hand erstellt werden.

Aus derselben Bearbeitungsumgebung entstehen zwei Arten von Torjubeln. Dynamische Ganzkörper tänze werden in Isaac Lab trainiert, damit eine Regelung die Bewegung in der Simulation kontrolliert ausführen kann. Einfachere Gesten für Kopf und Arme lassen sich ohne Training als Keyframe-Animationen nutzen. Beide Varianten wurden in die vorhandene Verhaltens- und Bewegungsarchitektur von B-Human integriert und können über Spielereignisse, Testbefehle oder einen Controller ausgelöst werden. Die Ausführung wurde abschließend auf dem realen K1 erprobt. Dabei entsprachen die sichtbaren Bewegungen den zuvor in der Simulation geprüften Abläufen.

Das Ergebnis ist ein zusammenhängender Arbeitsablauf vom Bewegungsvorbild bis zur Ausführung auf dem realen K1. Er stellt wiederverwendbare Schnittstellen für weitere Bewegungen bereit und verbindet Animation, Reinforcement Learning und die autonome Robotersteuerung.

Abstract

The transition from the smaller NAO robot to the larger Booster K1 changes the appearance of robot football. Its technical design appears more functional and less playful than that of its predecessor. This thesis explores how visible reactions and goal celebrations can give the robot a stronger sense of personality.

To support this goal, an end-to-end workflow was developed for creating, editing, and integrating dance motions into the B-Human system. The Macarena dance serves as a running example. A source video is reconstructed as human 3D motion and retargeted to the K1. A custom Blender add-on called K1 Animator then makes the motion visible and editable on a constrained robot rig. It can also be used to create new motions manually.

The workflow supports two types of celebrations. Dynamic full-body dances are trained in Isaac Lab so that a control policy can execute the reference motion in simulation. Simpler head and arm gestures are stored as keyframe animations and do not require training. Both variants are integrated into B-Human’s existing behavior and motion architecture and can be triggered by game events, test commands, or a controller. Their execution was subsequently tested on the physical K1, where the visible motions were consistent with the behavior previously observed in simulation.

The result is a coherent workflow from a motion reference to execution on the physical K1. It connects animation, reinforcement learning, and autonomous robot control through reusable interfaces for further motions.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
1.2	Ausgangslage	2
1.3	Ziel und Forschungsfrage	2
1.4	Eigener Beitrag und Vorgehensweise	3
1.5	Aufbau der Arbeit	3
2	Stand der Technik und Grundlagen	4
2.1	Booster K1	4
2.2	Bewegung aus einem Video	4
2.3	Keyframes und Tracking-Policies	5
2.4	B-Human als Zielsystem	6
2.5	Dateiformate der Pipeline	6
3	Konzept und End-to-End-Pipeline	7
3.1	Überblick	7
3.1.1	Quellcode und Repositorien	7
3.2	Teil I: Videobasierte Bewegungserfassung und Retargeting	7
3.2.1	Videovorlage	8
3.2.2	Rekonstruktion mit GVHMR	8
3.2.3	Retargeting mit GMR	9
3.3	Teil II: Bewegungserstellung, Training und Ausführung	10
3.3.1	Blender als Bearbeitungsstufe	10
3.3.2	Zwei Ausführungspfade	10
3.3.3	Kurzes Keyframe-Beispiel: Dab	11
3.3.4	Begründung des Aufbaus	12
4	Der K1 Animator	13
4.1	Ziel des Add-ons	13
4.2	K1-Rig	13
4.3	Import	14
4.4	Bearbeitung und manuelle Animation	15
4.5	Export	16
4.6	Typischer Ablauf	16
5	Training des Ganzkörperanzuges in Isaac Lab	17
5.1	Warum ein Training nötig ist	17
5.2	Trainingsumgebung	17
5.3	Beobachtung und Aktion	18
5.4	Lernziel	19
5.5	Trainingsablauf	20
5.6	Trainingsverlauf	20
5.7	Ergebnis und Export	21
6	Integration in B-Human	22
6.1	Übergabe aus dem Training	22
6.2	Architektur der Ausführung	23
6.3	EmoteEngine	23

6.4	Gemeinsame Laufzeiteinstellungen	24
6.5	Motorparameter während des Tanzes	24
6.6	Wiederholung und Übergänge	25
6.7	Keyframe-Pfad	25
6.8	PuppeteerEngine	26
6.9	Auslösung des Torjubels	26
6.10	Einordnung in die Sicherheitsstruktur	27
6.11	Zusammenspiel der Komponenten	27
7	Evaluation und Diskussion	28
7.1	Ziel der Evaluation	28
7.2	Prüfung des Bearbeitungs- und Exportablaufs	28
7.3	Schnittstelle zwischen Training und B-Human	28
7.4	Funktionale Prüfung in der Simulation	29
7.5	Emotes auf dem realen Roboter	30
7.6	Beantwortung der Forschungsfrage	30
7.7	Grenzen der Evaluation	30
7.8	Grenzen des Retargetings	31
8	Fazit und Ausblick	32
8.1	Fazit	32
8.2	Einordnung der gewählten Lösung	32
8.3	Nächste Schritte	33
	Literaturverzeichnis	34
A	Nutzung KI-basierter Anwendungen	35
A.1	Verwendete Werkzeuge	35
A.2	Art der Unterstützung	35
A.3	Eigenanteil und Verantwortung	35
B	Struktur einer Keyframe-Animation	36
C	Schnittstellenübersicht der Beispiele	37

Abbildungsverzeichnis

Abbildung 1	NAO und Booster K1 im Vergleich (eigene Darstellung)	1
Abbildung 2	Kinematisches Modell des Booster K1 mit seinen 22 Gelenken und deren Drehachsen; eigene Darstellung auf Grundlage des K1-Modells von BoosterRobotics (2026a), in Blender gerendert	4
Abbildung 3	Videobasierte Bewegungserfassung: GVHMR rekonstruiert die menschliche Bewegung, GMR überträgt sie auf die Kinematik des Booster K1 (eigene Darstellung)	7
Abbildung 4	Videovorlage des Macarena-Tanzes (eigene Videoaufnahme)	8
Abbildung 5	GVHMR-Rekonstruktion mit SMPL-Modell über der Videovorlage (links) und isoliertem SMPL-Modell (rechts) (eigener Screenshot aus der GVHMR-Ausgabe auf Grundlage der eigenen Videoaufnahme)	8
Abbildung 6	Mit GMR auf die Kinematik des Booster K1 übertragene Macarena-Bewegung (eigener Screenshot aus der GMR-Ausgabe auf Grundlage der eigenen Videoaufnahme)	9
Abbildung 7	Bewegungs-zu-Roboter-Pipeline: Blender bearbeitet die Referenzbewegung, Isaac Lab trainiert bei Bedarf eine Policy und B-Human führt sie aus (eigene Darstellung)	10
Abbildung 8	Vom Dab-Video zur ausgeführten Keyframe-Animation: Videovorlage, Bearbeitung im K1 Animator und Ergebnis in SimRobot (eigene Aufnahme und eigene Screenshots)	11
Abbildung 9	Bearbeitung der Macarena-Bewegung in Blender mit K1-Rig, Bones und Gelenkgrenzen sowie K1 Animator und NLA-Editor (eigener Screenshot)	13
Abbildung 10	K1-Rig mit ausgewählten IK-Zielen für Hände und Füße während der manuellen Posenbearbeitung (eigener Screenshot)	14
Abbildung 11	Erzeugung des K1-Rigs und Import einer Bewegungsdatei im K1 Animator (eigene Screenshots)	15
Abbildung 12	Animationsbereich des K1 Animator mit Optionen für NLA- und Graph-Editor sowie Keyframe-Glättung (eigener Screenshot)	15
Abbildung 13	Exportbereich des K1 Animator mit den verfügbaren Zielformaten (eigener Screenshot)	16
Abbildung 14	Booster K1 im Isaac-Sim-Viewport, der Physik-Engine hinter Isaac Lab (eigener Screenshot)	17
Abbildung 15	Mittlere Episodenbelohnung bis Trainingsiteration 11.026 (eigene Darstellung auf Grundlage der TensorBoard-Trainingsdaten)	20
Abbildung 16	Mittlere Episodenlänge bis Trainingsiteration 11.026 (eigene Darstellung auf Grundlage der TensorBoard-Trainingsdaten)	21
Abbildung 17	Ausführung des Macarena-Tanzes durch die EmoteEngine in SimRobot (eigener Screenshot)	22

Tabellenverzeichnis

Tabelle 1	Vereinfachte Einordnung des Torjubels in die B-Human-Architektur	6
Tabelle 2	Zentrale Dateiformate und ihre Rolle in der Pipeline	6
Tabelle 3	Die beiden Ausführungspfade nach der Bearbeitung in Blender	10
Tabelle 4	Exportformate des K1 Animator	16
Tabelle 5	Aktuator- und Aktionsparameter des verwendeten Macarena- Trainingslaufs	18
Tabelle 6	Zusammensetzung der Policy-Eingabe	18
Tabelle 7	Reward-Gruppen und Gewichte des Macarena-Trainings	19
Tabelle 8	Datenfluss vom Spielereignis bis zur Bewegung	23
Tabelle 9	Die beiden Ausführungsarten der EmoteEngine	24
Tabelle 10	Geprüfte Schnittstellen zwischen Training und Laufzeit	29
Tabelle 11	Funktionale Anforderungen und ihre Nachweise	29
Tabelle 12	Struktur einer Keyframe-Animation für die EmoteEngine	36
Tabelle 13	Keyframe-Emotes im aktuellen B-Human-Konfigurationsstand	36
Tabelle 14	Dateien und Schnittstellen der durchlaufenden Beispiele	37

Abkürzungsverzeichnis

CFG	Konfigurationsdatei im B-Human-Format
CSV	Comma-Separated Values
ELU	Exponential Linear Unit
FK	Forward Kinematics
GMR	General Motion Retargeting
GVHMR	World-Grounded Human Motion Recovery via Gravity-View Coordinates
HSL	Humanoid Soccer League
IK	Inverse Kinematics
NLA	Nonlinear Animation
NPZ	komprimiertes NumPy-Dateiformat
ONNX	Open Neural Network Exchange
PD	Proportional-Derivative (Regelung)
PKL	Python-Pickle-Datei
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
SMPL	Skinned Multi-Person Linear Model
SPL	Standard Platform League
URDF	Unified Robot Description Format

1 Einleitung

1.1 Motivation

Der Roboterfußball befindet sich in einem sichtbaren Wandel. B-Human spielte über viele Jahre mit NAO-Robotern in der Standard Platform League (SPL). Seit die SPL und die frühere Humanoid League zur Humanoid Soccer League (HSL) zusammengeführt wurden, tritt das Team dort mit dem humanoiden Booster K1 an (B-Human, 2026; RoboCup Federation, 2026). Der Wechsel eröffnet technisch viele Möglichkeiten, verändert aber auch den Charakter der Liga.

Der NAO prägte den RoboCup über viele Jahre. Seine runden Formen, die geringe Größe und seine oft etwas unbeholfen wirkenden Bewegungen gaben ihm einen hohen Wiedererkennungswert. Der Booster K1 wirkt dagegen größer, technischer und funktionaler (Abbildung 1). Genau daraus entstand die Motivation für diese Arbeit: Der neue Roboter sollte nicht nur Aufgaben auf dem Spielfeld erfüllen, sondern auch sichtbar reagieren können. Ein Torjubiläum bringt Bewegung, Emotion und einen kleinen Spaßfaktor in eine Situation, die für Zuschauer ohnehin leicht verständlich ist.

Die Arbeit verfolgt damit kein rein dekoratives Ziel. Ein autonom ausgelöster Jubel verbindet die äußere Wirkung des Roboters mit seiner internen Spielentscheidung. Gleichzeitig entsteht ein praktischer Anwendungsfall, an dem sich verschiedene technische Bereiche zusammenführen lassen: Motion Capture, Animation, Reinforcement Learning und die Bewegungssteuerung von B-Human.

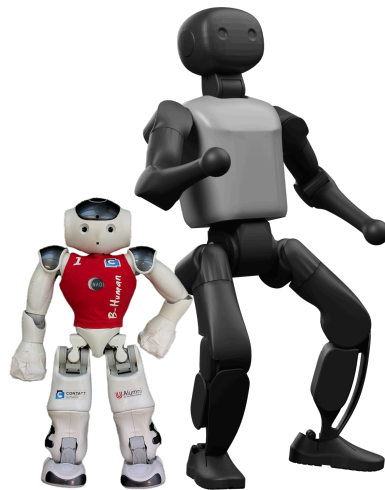


Abbildung 1: NAO und Booster K1 im Vergleich (eigene Darstellung)

1.2 Ausgangslage

Eine Tanzbewegung lässt sich nicht direkt aus einem Video auf den K1 übertragen. Mensch und Roboter besitzen unterschiedliche Proportionen, Gelenke und Bewegungsgrenzen. Nach dem Retargeting muss die Bewegung deshalb kontrolliert und bei Bedarf angepasst werden. Ein rein textbasierter Workflow ist dafür unpraktisch, weil sich aus langen Zahlenfolgen kaum erkennen lässt, wie die Bewegung am Roboter aussieht und an welchen Stellen sie angepasst werden muss.

Für dynamische Ganzkörperbewegungen kommt eine weitere Aufgabe hinzu. Eine Animation beschreibt zunächst nur, wie eine Pose aussehen soll. Sie reagiert nicht auf den aktuellen Zustand des Roboters. Für den Macarena-Tanz wird deshalb eine Tracking-Policy trainiert, die der Referenzbewegung in der Physiksimulation folgt. Einfachere Bewegungen, bei denen die Beine im Stand bleiben, können weiterhin als Keyframes abgespielt werden.

Am Ende müssen beide Varianten in B-Human ankommen und damit auch auf dem realen Roboter ausführbar sein. Dort darf ein Torjubel kein isoliertes Demo-Programm sein, sondern muss sich in die bestehende Verhaltens- und Bewegungssteuerung einordnen. Nur so kann er durch ein Spielereignis ausgelöst und durch die vorhandenen Stoppmechanismen wieder beendet werden.

1.3 Ziel und Forschungsfrage

Ziel der Arbeit ist ein zusammenhängender Workflow, mit dem performative Bewegungen für den Booster K1 erstellt und als Torjubel in B-Human genutzt werden können. Im Mittelpunkt steht folgende Forschungsfrage:

„Wie lässt sich eine durchgängige Pipeline entwickeln, die Tanzbewegungen für den Booster K1 aus einer Vorlage erzeugt, visuell bearbeitbar macht und als ereignisgesteuerten Torjubel in das B-Human-System integriert?“

Daraus ergeben sich drei Teilfragen:

1. Wie kann eine menschliche Bewegung auf die Kinematik des K1 übertragen und anschließend am Robotermodell bearbeitet werden?
2. Wie lassen sich bearbeitete Bewegungen je nach Umfang und Dynamik als trainierter Ganzkörper Tanz oder als direkte Keyframe-Animation ausführen?
3. Wie müssen diese Bewegungen in die Architektur von B-Human eingebunden werden, damit sie über reguläre Verhaltensentscheidungen ausgeführt werden können?

Der technische Schwerpunkt liegt auf dem entwickelten K1 Animator, der Trainings- und Datenpipeline sowie der B-Human-Integration. Der Macarena-Tanz dient als roter Faden für den Policy-Pfad. Die kurze Jubelbewegung „Dab“ ergänzt ihn als kompaktes Beispiel für den direkten Keyframe-Pfad.

1.4 Eigener Beitrag und Vorgehensweise

Der K1 Animator für Blender und die Integration der Bewegungen in B-Human bilden die beiden zentralen eigenen Beiträge. Das Add-on stellt ein animierbares Modell des K1 bereit, importiert Bewegungsdaten und exportiert sie in die Formate der nachfolgenden Systeme. Damit dient es innerhalb der entwickelten Pipeline als gemeinsame visuelle Arbeitsoberfläche für retargetete und manuell erstellte K1-Bewegungen.

Darauf aufbauend wurde die Trainingsumgebung für einen Ganzkörper Tanz eingerichtet und mit der B-Human-Laufzeit verbunden. Zusätzlich entstand ein Keyframe-Pfad für einfachere Gesten. Auf der B-Human-Seite verwaltet die dafür entwickelte EmoteEngine beide Tanzarten und bindet sie in den normalen Weg von der Spielentscheidung bis zur Gelenkausgabe ein. Die ebenfalls in B-Human eingebundene PuppeteerEngine ergänzt diesen Ablauf um ein SimRobot-Werkzeug, mit dem vorhandene Keyframe-Bewegungen geladen, abgespielt und einzelne Posen manuell geprüft werden können. Das Erzeugen und Speichern neuer Sequenzen übernimmt der K1 Animator.

Die Evaluation greift diese drei Teilfragen wieder auf. Erstens wird geprüft, ob übertragene Bewegungen im K1 Animator am Robotermodell bearbeitet und in die benötigten Formate ausgegeben werden können. Zweitens werden der trainierte Policy-Pfad und der direkte Keyframe-Pfad auf ihre jeweils benötigten Schnittstellen und ihre Bewegungsausgabe untersucht. Drittens wird geprüft, ob beide Varianten über die reguläre B-Human-Architektur in SimRobot und auf dem realen K1 ausgelöst und ausgeführt werden können.

Der in dieser Arbeit betrachtete K1 Animator liegt als installierbares Add-on-Paket in Version 1.0.0 vor. Für die Bewertung des Policy-Pfads sind die aufgelöste Trainingskonfiguration und das exportierte ONNX-Modell maßgeblich.

1.5 Aufbau der Arbeit

Kapitel 2 stellt die Grundlagen vor, die für den weiteren Ablauf benötigt werden. Kapitel 3 zeigt den Datenfluss am Beispiel des Macarena-Tanzes. Kapitel 4 beschreibt den K1 Animator als zentrales Bearbeitungswerkzeug. Danach folgen das Policy-Training in Kapitel 5 und die Integration in B-Human in Kapitel 6. Kapitel 7 bewertet die umgesetzten Beispiele und ihre Schnittstellen. Kapitel 8 fasst die Ergebnisse zusammen und beschreibt die nächsten Schritte.

2 Stand der Technik und Grundlagen

Dieses Kapitel führt die Begriffe ein, die für den weiteren Ablauf benötigt werden. Es behandelt den K1 als verwendeten Roboter, die Gewinnung einer Bewegung aus Video, die beiden Arten der späteren Ausführung und die Grundidee der B-Human-Architektur.

2.1 Booster K1

Der Booster K1 ist ein humanoider Roboter mit 22 angetriebenen Gelenken. Sie verteilen sich auf Kopf, Arme und Beine (BoosterRobotics, 2026a). In den verwendeten Bewegungsdateien wird eine Pose deshalb im Kern durch 22 Gelenkwinkel und die Lage des Rumpfes beschrieben (Abbildung 2).

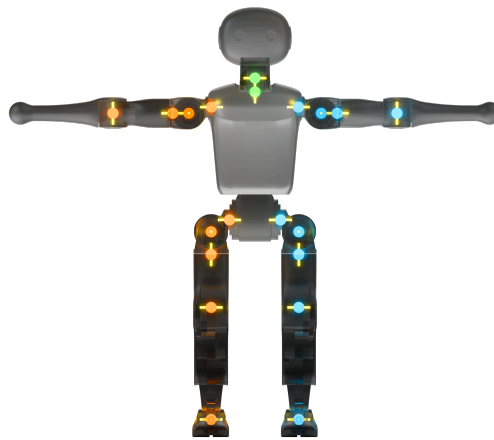


Abbildung 2: Kinematisches Modell des Booster K1 mit seinen 22 Gelenken und deren Drehachsen; eigene Darstellung auf Grundlage des K1-Modells von BoosterRobotics (2026a), in Blender gerendert

Für die Pipeline ist nicht nur die Zahl der Gelenke wichtig. Jedes Gelenk besitzt eine feste Drehachse und einen begrenzten Bewegungsbereich. Eine menschliche Pose kann daher nicht unverändert übernommen werden. Das Robotermodell muss entscheiden, welche K1-Pose der beabsichtigten Bewegung möglichst nahekommt.

Die geometrische Struktur des Roboters wird in einer URDF-Datei (Unified Robot Description Format) beschrieben. Sie legt fest, wie die einzelnen Körperteile und Gelenke miteinander verbunden sind. Der K1 Animator und Isaac Lab nutzen diese Beschreibung an unterschiedlichen Stellen, etwa für die Berechnung der Körperposen und für die Physiksimulation.

2.2 Bewegung aus einem Video

Klassisches Motion Capture arbeitet häufig mit mehreren Kameras oder Markern am Körper. Für den hier geplanten Workflow wäre dieser Aufbau unnötig aufwendig. Stattdessen wird eine monokulare Videoaufnahme verwendet. Sie lässt sich schnell erstellen und erlaubt es, vorhandene Tanzvorlagen als Ausgangspunkt zu nutzen.

GVHMR rekonstruiert aus dem Video eine menschliche Bewegung im dreidimensionalen Raum (Shen et al., 2024). Dabei wird nicht nur die Körperhaltung in jedem Bild

geschätzt. Das Verfahren berücksichtigt auch Schwerkraft und Kamerarichtung, damit aus den Einzelbildern eine zusammenhängende Bewegung im Raum entsteht. Die Körperpose basiert auf SMPL, einem parametrischen Modell des menschlichen Körpers (Loper et al., 2015). Das Ergebnis beschreibt weiterhin einen Menschen und noch keinen Roboter.

Diese Übersetzung übernimmt GMR (Araujo et al., 2025). Das Verfahren retargetet die menschliche Bewegung auf ein vorgegebenes Robotermodell. Dafür vergleicht es ausgewählte Körperpunkte von Mensch und Roboter und passt die Roboterpose schrittweise an die Vorlage an. Dabei werden die abweichenden Proportionen, Gelenkachsen und Freiheitsgrade berücksichtigt. Im Projekt entsteht daraus eine PKL-Datei mit der Root-Bewegung und den 22 K1-Gelenkwinkeln.

GVHMR und GMR wurden gewählt, weil sie eine vollständige Bewegung aus einer gewöhnlichen Videoaufnahme bis in den Gelenkraum des K1 überführen. Eine vollständige Handanimation bleibt trotzdem möglich. Sie ist vor allem dann sinnvoll, wenn keine geeignete Videovorlage existiert oder nur eine kurze, gezielt entworfene Geste benötigt wird.

2.3 Keyframes und Tracking-Policies

Eine Keyframe-Animation gibt zu bestimmten Zeitpunkten feste Gelenkstellungen vor. Zwischen diesen Posen wird interpoliert. Dieses Prinzip ist übersichtlich und eignet sich gut für Kopf- und Armbewegungen, während die Beine in einer stabilen Standbewegung bleiben.

Bei einem dynamischen Ganzkörper Tanz reicht eine solche offene Folge von Sollwerten nicht immer aus. Der Roboterzustand kann von der Referenz abweichen, ohne dass die Animation darauf reagiert. Für diesen Fall wird eine Tracking-Policy trainiert. Sie erhält neben dem aktuellen Referenzframe auch Informationen über den Zustand des simulierten Roboters und berechnet daraus die nächsten Gelenkziele.

Beim Reinforcement Learning lernt eine Policy durch wiederholte Interaktion mit einer Umgebung. In jedem Schritt erhält sie eine Beobachtung, wählt eine Aktion und bekommt anschließend eine Belohnung. Während des Trainings werden ihre Parameter so angepasst, dass über viele Durchläufe eine möglichst hohe kumulierte Belohnung erreicht wird (Sutton & Barto, 2018). Beim Motion Tracking fällt diese Belohnung höher aus, wenn der Roboter der Referenzbewegung genau folgt; zusätzliche Anteile können unerwünschte Bewegungen oder Kontakte bestrafen.

Für die Aktualisierung der Policy wird Proximal Policy Optimization (PPO) verwendet (Schulman et al., 2017). PPO lernt aus den zuvor gesammelten Simulationsschritten und begrenzt, wie stark sich die Policy bei einer einzelnen Aktualisierung verändern darf. Dadurch soll verhindert werden, dass bereits erlerntes Verhalten durch einen zu großen Optimierungsschritt sofort wieder verloren geht. Die Grundidee der Bewegungsnachahmung folgt Arbeiten wie DeepMimic (Peng, Abbeel, et al., 2018). Die verwendete K1-Umgebung orientiert sich am BeyondMimic-Ansatz und wird in Isaac Lab umgesetzt (BoosterRobotics, 2026b; Liao et al., 2025). Die Referenz gibt dabei weiterhin den Tanz vor; die Policy lernt seine zustandsabhängige Ausführung.

2.4 B-Human als Zielsystem

B-Human ist modular aufgebaut und trennt Wahrnehmung, Verhaltensentscheidung und Bewegungserzeugung (Röfer et al., 2024; Röfer et al., 2025). Informationen werden in sogenannten Representations abgelegt. Modules lesen benötigte Representations und stellen neue bereit. Dadurch bleiben die einzelnen Aufgaben voneinander getrennt.

Für einen Torjubil sind drei Ebenen besonders wichtig (Tabelle 1):

Ebene	Aufgabe	Bezug zum Torjubil
Strategy	Verhalten auswählen	erkennt ein Ereignis und fordert einen Jubel an
Skill	Anforderung konkretisieren	übersetzt die Entscheidung in einen Bewegungswunsch
Motion	Gelenkziele erzeugen	führt Policy- oder Keyframe-Tanz aus

Tabelle 1: Vereinfachte Einordnung des Torjubels in die B-Human-Architektur

In dieser Struktur setzt die Torerkennung nicht direkt Gelenkwinkel. Die Verhaltenssteuerung entscheidet lediglich, dass gefeiert werden soll. Erst der Motion-Thread erzeugt daraus eine konkrete Bewegung. Der Jubel kann so dieselben Übergänge und Abbruchmechanismen verwenden wie andere Bewegungen des Roboters.

2.5 Dateiformate der Pipeline

Die Pipeline verbindet mehrere Programme, die jeweils eigene Formate verwenden. Tabelle 2 zeigt den für diese Arbeit verwendeten Datenfluss.

Format	Stufe	Inhalt und Verwendung
.mp4	Video	aufgenommene menschliche Tanzvorlage
.pt	GVHMR	rekonstruierte menschliche Bewegung
.pkl	GMR	auf den K1 retargetete Root- und Gelenkbewegung
.csv	K1 Animator	Isaac-Bewegungsdaten aus Root-Pose und Gelenkwinkeln
.npz	K1 Animator / Isaac Lab	vollständige Referenzbewegung für das Policy-Training
.onnx	B-Human	trainiertes Netz für den Ganzkörperanz
.cfg	K1 Animator / B-Human	Referenztrajektorie der Policy oder eigenständige Keyframe-Animation

Tabelle 2: Zentrale Dateiformate und ihre Rolle in der Pipeline

Der K1 Animator kann PKL-, CSV-, NPZ- und CFG-Dateien importieren und CSV, NPZ sowie beide CFG-Varianten exportieren. Für den Policy-Pfad entstehen in Blender eine NPZ-Trainingsreferenz und eine inhaltlich passende CFG-Referenztrajektorie. Das Training ergänzt das ONNX-Modell. Ein Keyframe-Tanz benötigt dagegen nur seine Animations-CFG.

3 Konzept und End-to-End-Pipeline

3.1 Überblick

Der entwickelte Workflow führt von einer Bewegungsvorlage bis zu einem ausführbaren Torjubel. Der Macarena-Tanz dient dabei als zusammenhängendes Beispiel. Jede Stufe hat eine klar abgegrenzte Aufgabe: Bewegung erfassen, auf den K1 übertragen, bearbeiten, gegebenenfalls trainieren und schließlich in B-Human ausführen.

Die Pipeline gliedert sich in zwei Teile. Der erste Teil erzeugt aus der menschlichen Videovorlage eine K1-kompatible Referenzbewegung. Der zweite Teil macht diese Bewegung bearbeitbar, trainiert bei Bedarf eine Policy und führt sie in B-Human aus.

3.1.1 Quellcode und Repositorien

Die eigenen Softwareanteile stehen der Arbeit als veröffentlichtes beziehungsweise zugangsbeschränktes Repository zur Verfügung.

K1 Animator Blender-Add-on, Version 1.0.0 gitlab.informatik.uni-bremen.de/postels/k1-animator 	B-Human-Integration Branch <code>p/postels/thesis</code> (zugangsbeschränkt) B-Human / p/postels/thesis (intern) 
--	---

3.2 Teil I: Videobasierte Bewegungserfassung und Retargeting

Abbildung 3 zeigt den ersten Teil der Pipeline. Die Videoaufnahme wird zunächst als menschliche Bewegung rekonstruiert und anschließend auf die Kinematik des K1 übertragen.



Abbildung 3: Videobasierte Bewegungserfassung: GVHMR rekonstruiert die menschliche Bewegung, GMR überträgt sie auf die Kinematik des Booster K1 (eigene Darstellung)

3.2.1 Videovorlage

Am Anfang steht eine Videoaufnahme des Macarena-Tanzes. Sie dient nicht als direkt abspielbare Animation, sondern als visuelles Bewegungsvorbild. Der ausgewählte Ausschnitt sollte die Person möglichst vollständig zeigen, damit Arme, Beine und Rumpf über den gesamten Tanz hinweg erkennbar bleiben (Abbildung 4).

Die Videovorlagen wurden für diese Arbeit selbst aufgenommen. Die abgebildeten Personen haben der Aufnahme und ihrer Verwendung in dieser Arbeit schriftlich zugestimmt.



Abbildung 4: Videovorlage des Macarena-Tanzes (eigene Videoaufnahme)

3.2.2 Rekonstruktion mit GVHMR

GVHMR verarbeitet die MP4-Datei und schätzt daraus eine menschliche Bewegung im Raum. Die Ausgabe wird als PT-Datei gespeichert. Für den weiteren Ablauf ist entscheidend, dass nicht nur einzelne 2D-Punkte entstehen, sondern eine zeitlich zusammenhängende 3D-Bewegung mit einer globalen Ausrichtung.

Die Rekonstruktion kann gemeinsam mit dem Originalvideo betrachtet werden. Dadurch lässt sich früh erkennen, ob die Körperhaltung und der zeitliche Verlauf grundsätzlich passen (Abbildung 5). An dieser Stelle wird noch keine K1-Bewegung erzeugt.

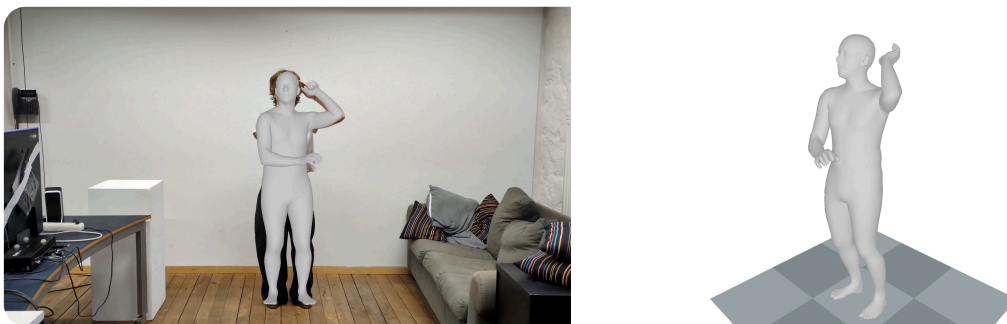


Abbildung 5: GVHMR-Rekonstruktion mit SMPL-Modell über der Videovorlage (links) und isoliertem SMPL-Modell (rechts) (eigener Screenshot aus der GVHMR-Ausgabe auf Grundlage der eigenen Videoaufnahme)

3.2.3 Retargeting mit GMR

GMR übernimmt die rekonstruierte menschliche Bewegung und überträgt sie auf die Gelenkstruktur des K1. Das Ergebnis ist eine PKL-Datei mit der Bewegung des Rumpfes und den 22 Gelenkwinkeln des Roboters (Abbildung 6). Ab diesem Schritt liegt die Bewegung in einer K1-spezifischen Repräsentation vor.

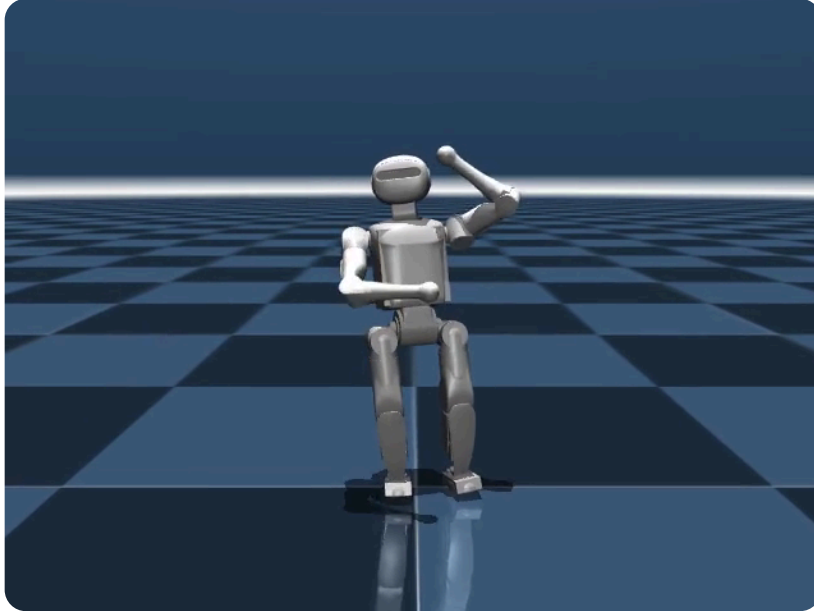


Abbildung 6: Mit GMR auf die Kinematik des Booster K1 übertragene Macarena-Bewegung (eigener Screenshot aus der GMR-Ausgabe auf Grundlage der eigenen Videoaufnahme)

Das Retargeting ist mehr als ein direktes Bone-Mapping. Ein Mensch besitzt andere Proportionen und Bewegungsmöglichkeiten als der K1. GMR sucht deshalb eine passende Roboterbewegung, die die Vorlage unter den Grenzen des Zielmodells möglichst gut wiedergibt. Für die gestalterische Beurteilung bleibt eine visuelle Nachbearbeitung sinnvoll.

3.3 Teil II: Bewegungserstellung, Training und Ausführung

Abbildung 7 zeigt den zweiten Teil der Pipeline. Er beginnt mit der K1-spezifischen Referenzbewegung aus GMR und endet mit einer auslösbaren Bewegung in B-Human.

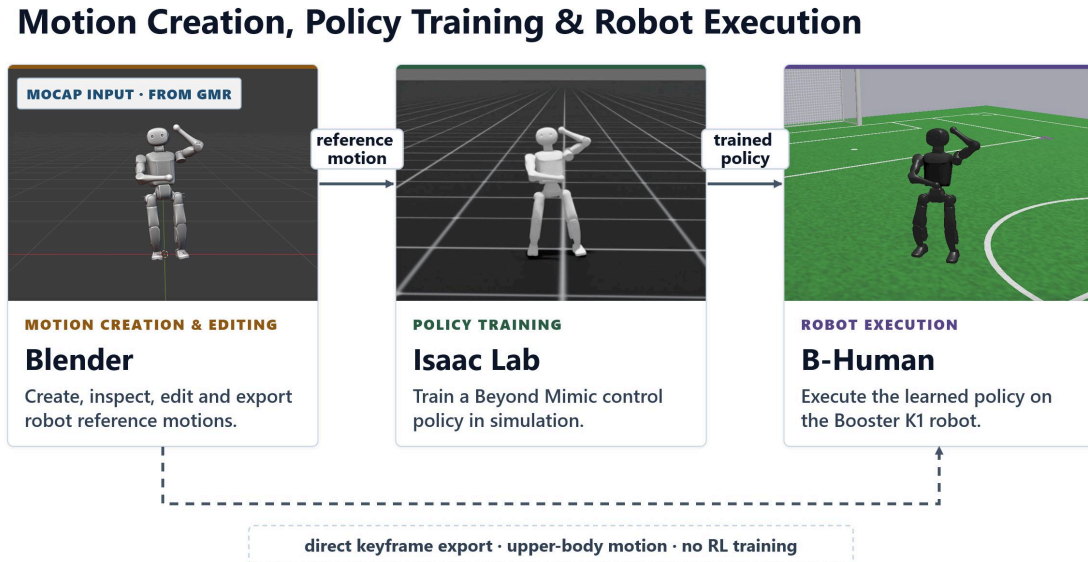


Abbildung 7: Bewegungs-zu-Roboter-Pipeline: Blender bearbeitet die Referenzbewegung, Isaac Lab trainiert bei Bedarf eine Policy und B-Human führt sie aus (eigene Darstellung)

Blender bildet den Auftakt dieses Teils, weil dort eine visuelle und manuell veränderbare K1-Bewegung vorliegt. Ab diesem Punkt verzweigt der Workflow in einen Policy-Pfad für dynamische Ganzkörperbewegungen und einen Keyframe-Pfad für einfachere Gesten.

3.3.1 Blender als Bearbeitungsstufe

Die PKL-Datei wird in den K1 Animator importiert. Das Add-on rekonstruiert daraus eine Blender-Animation auf dem K1-Rig. Der Bewegungsablauf kann nun auf einer Timeline abgespielt, an einzelnen Gelenken verändert oder vollständig neu aufgebaut werden.

Blender ist bewusst als optionale Bearbeitungsstufe ausgelegt. Eine bereits geeignete Bewegung kann ohne größere Änderungen weiterverarbeitet werden. Für den Macarena-Tanz ist die Blender-Stufe dennoch wichtig, weil sie die gemeinsame Sicht auf Bewegung, Gelenkgrenzen und spätere Ausgabeformate bietet.

3.3.2 Zwei Ausführungspfade

Nach der Bearbeitung stehen zwei Wege zur Verfügung (Tabelle 3):

Pfad	Einsatz	Ergebnis für B-Human
Policy-Tanz	dynamische Ganzkörperbewegung mit Training in Isaac Lab	ONNX-Modell und CFG-Referenztrajektorie
Keyframe-Tanz	direkte Kopf- und Armbewegung ohne Training	Animations-CFG

Tabelle 3: Die beiden Ausführungspfade nach der Bearbeitung in Blender

Der Policy-Pfad wird für den Macarena-Tanz verwendet. Blender liefert dafür eine NPZ-Referenz für Isaac Lab und die dazugehörige CFG-Referenztrajektorie für B-Human. Isaac Lab trainiert aus der NPZ-Bewegung die Tracking-Policy und ergänzt damit das ONNX-Modell für die Ausführung.

Der Keyframe-Pfad überspringt das Training. Er eignet sich für kurze Gesten, bei denen der vorhandene StandGenerator weiterhin die Beine kontrolliert. Beide Wege beginnen im selben Bearbeitungswerkzeug, unterscheiden sich aber deutlich in Aufwand und Einsatzbereich.

BEGLEITVIDEO

VIDEO 1 - DIE MACARENA-PIPELINE: VON DER VIDEOVORLAGE BIS ZUM ROBOTER

Link: youtu.be/F-60jKfk3OU



3.3.3 Kurzes Keyframe-Beispiel: Dab

Der Dab dient in dieser Arbeit als Beispiel für den kürzeren Ausführungspfad. Die Bewegung wird im K1 Animator als Keyframe-Animation aufbereitet, als CFG-Datei exportiert und anschließend in B-Human eingebunden. Die daraus entstandene Datei `keyframe_dab.cfg` umfasst 93 Frames mit einer Animationsdauer von 3.069 ms. Die EmoteEngine übernimmt daraus Kopf und Arme, während der StandGenerator die Beine führt. Da keine Ganzkörper-Policy benötigt wird, entfällt das Training in Isaac Lab vollständig.

Der kompakte Ablauf ist in Abbildung 8 zusammengefasst.

Dab: Direct Keyframe Workflow

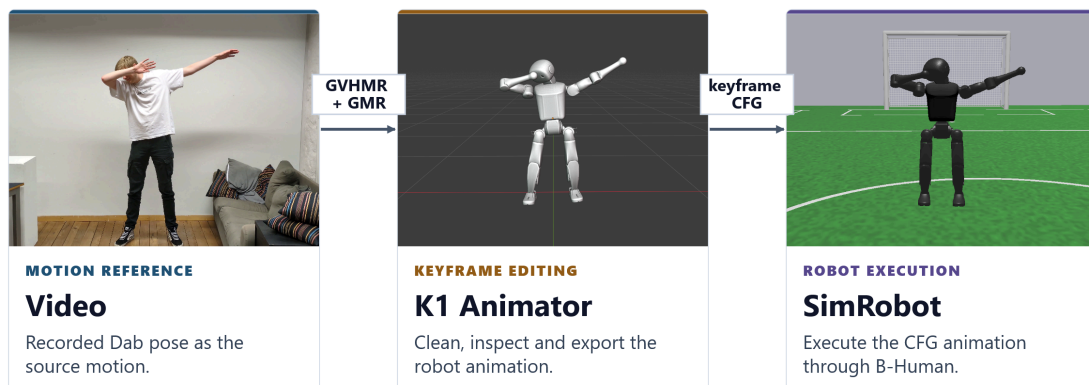


Abbildung 8: Vom Dab-Video zur ausgeführten Keyframe-Animation: Videovorlage, Bearbeitung im K1 Animator und Ergebnis in SimRobot (eigene Aufnahme und eigene Screenshots)

BEGLEITVIDEO

VIDEO 2 - VOM DAB-VIDEO ZUR KEYFRAME-ANIMATION IN B-HUMAN

Link: youtu.be/h86fMNLqFRQ



3.3.4 Begründung des Aufbaus

Eine vollständig manuelle Animation wäre für kurze Gesten ausreichend, wird bei längeren Tänzen aber schnell aufwendig. Der Videopfad liefert dagegen zügig eine zusammenhängende Ausgangsbewegung. Blender ergänzt die automatische Pipeline um die notwendige menschliche Kontrolle. Das Policy-Training wird erst dort eingesetzt, wo eine zeitabhängige Ganzkörperregelung benötigt wird.

Diese Aufteilung hält die einzelnen Werkzeuge austauschbar. Eine andere Videovorlage kann denselben Weg durchlaufen, eine manuell erstellte Blender-Animation kann GVHMR und GMR überspringen, und ein einfacher Keyframe-Tanz wie der Dab benötigt keine Trainingsumgebung.

4 Der K1 Animator

4.1 Ziel des Add-ons

Der K1 Animator ist ein im Rahmen dieser Arbeit entwickeltes Blender-Add-on und das zentrale Entwicklungswerkzeug der Pipeline. Es ist für Blender 5.2 oder neuer ausgelegt und macht den Booster K1 direkt in Blender animierbar (Blender Foundation, 2026). Es verbindet die Daten aus Motion Capture und Retargeting mit den Formaten von Isaac Lab und B-Human.

Das Werkzeug ist nicht auf Torjubel beschränkt. Es kann importierte Bewegungen bearbeiten, neue Animationen von Grund auf erstellen und den K1 für Renderings oder Bewegungstests darstellen. Für den Workflow dieser Arbeit ist besonders wichtig, dass dieselbe Blender-Animation in mehrere Zielformate exportiert werden kann.

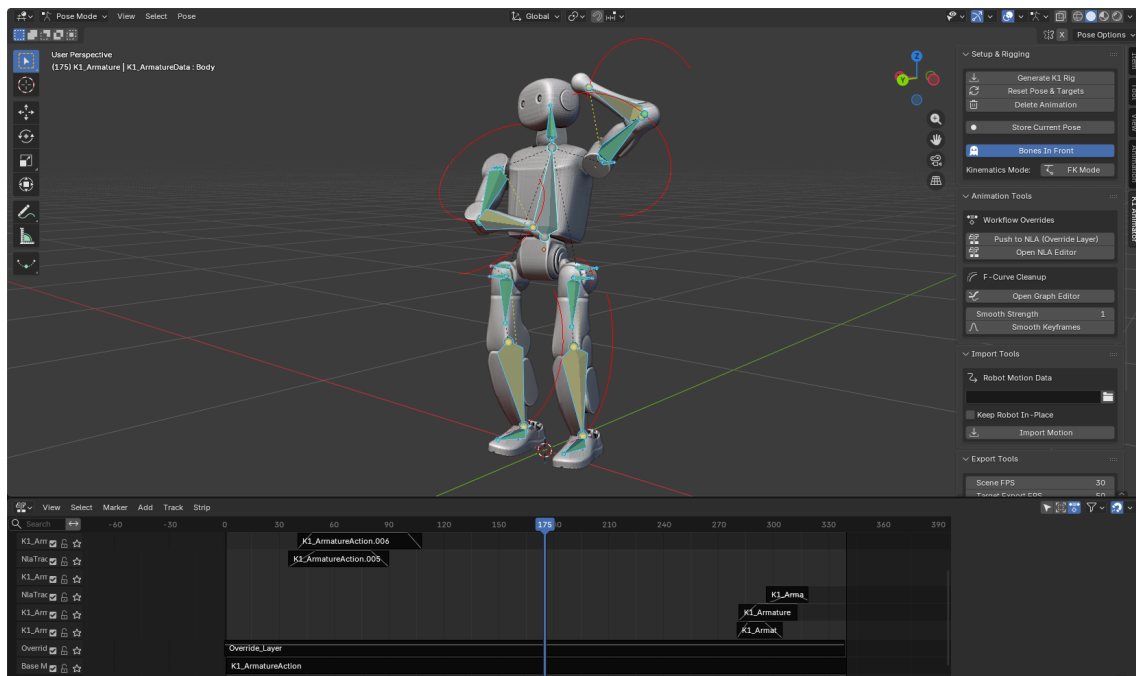


Abbildung 9: Bearbeitung der Macarena-Bewegung in Blender mit K1-Rig, Bones und Gelenkgrenzen sowie K1 Animator und NLA-Editor (eigener Screenshot)

Die Oberfläche gliedert sich in vier Bereiche (Abbildung 9). Im Setup wird das K1-Rig erzeugt. Die Animationswerkzeuge unterstützen das Erstellen und Bearbeiten von Keyframes. Der Import liest vorhandene Bewegungen ein. Im Export wird ausgewählt, ob die Animation für Isaac Lab, B-Human oder als einfaches Zwischenformat ausgegeben werden soll.

4.2 K1-Rig

Das Add-on erzeugt auf Knopfdruck eine Armature mit einem Root und 22 beweglichen Gelenken. Jedes Gelenk entspricht einem Freiheitsgrad des K1. Rotationsachsen, gesperrte Rotationsrichtungen und Gelenkgrenzen sind im Rig hinterlegt, damit Posen bereits während der Animation zur Kinematik des Roboters passen.

Die sichtbaren Meshes stammen aus dem K1-Modell von B-Human. Dadurch zeigt Blender denselben Roboter, der später auch in SimRobot verwendet wird. Das Modell dient in Blender als visuelle Referenz; die eigentliche Physiksimulation findet weiterhin in Isaac Lab beziehungsweise SimRobot statt.

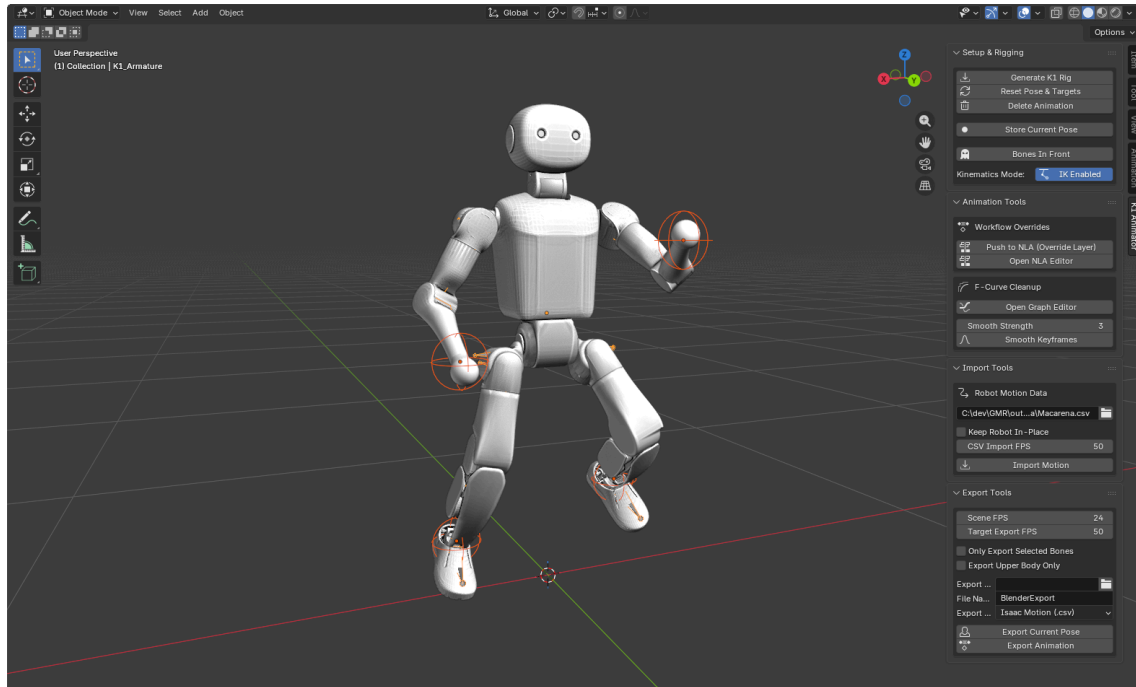


Abbildung 10: K1-Rig mit ausgewählten IK-Zielen für Hände und Füße während der manuellen Posenbearbeitung (eigener Screenshot)

Für Hände und Füße stehen IK-Ziele zur Verfügung (Abbildung 10). Statt jedes Gelenk einzeln zu drehen, kann beispielsweise die gewünschte Handposition direkt im Raum verschoben werden. Blender berechnet daraus eine passende Armhaltung. Für Arme und Beine kann jederzeit zwischen FK und IK gewechselt werden. Beim Speichern einer Pose überführt das Add-on das sichtbare IK-Ergebnis wieder in feste Gelenkwerte, die als Keyframes bearbeitet und exportiert werden können.

4.3 Import

Der K1 Animator unterstützt PKL-, CSV-, NPZ- und CFG-Dateien. Im Hauptpfad des Macarena-Tanzes wird die PKL-Ausgabe von GMR importiert. Das Add-on liest die Root-Bewegung und die 22 Gelenkwinkel und erzeugt daraus eine Blender-Action (Abbildung 11).

Die weiteren Formate erleichtern den Austausch mit bestehenden Werkzeugen. Eine CSV enthält Root-Pose und Gelenkwinkel in einer einfachen tabellarischen Form. Eine NPZ kann eine bereits für Isaac Lab aufbereitete Referenzbewegung zurück nach Blender bringen. Bei CFG-Dateien erkennt das Add-on selbstständig, ob eine Keyframe-Animation oder eine Referenztrajektorie für einen Policy-Tanz vorliegt.

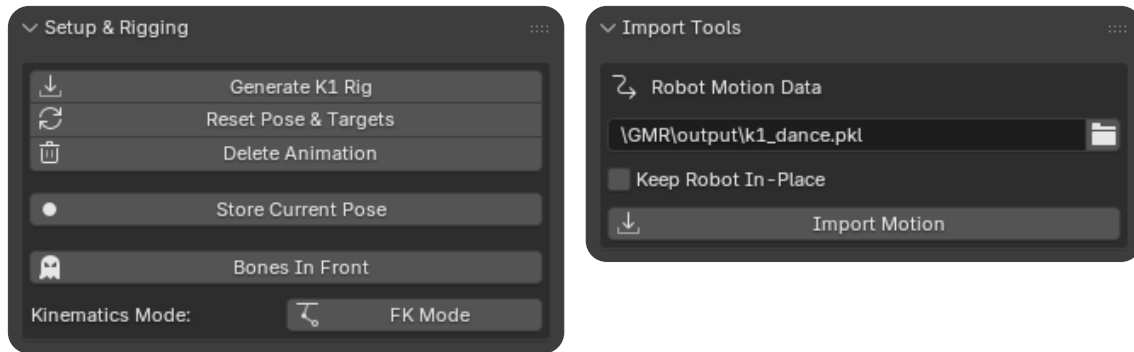


Abbildung 11: Erzeugung des K1-Rigs und Import einer Bewegungsdatei im K1 Animator (eigene Screenshots)

Beim Import übersetzt das Add-on zwischen den unterschiedlichen Gelenknamen und Koordinatensystemen der beteiligten Programme. Diese Übersetzung bleibt für die Person am Werkzeug weitgehend unsichtbar: Nach dem Import liegt eine normale Blender-Animation vor.

4.4 Bearbeitung und manuelle Animation

Die Bewegung kann auf der Timeline abgespielt und an jedem Frame verändert werden. Einzelne Gelenke lassen sich direkt drehen; Hände und Füße können über IK positioniert werden. Der Graph Editor zeigt den zeitlichen Verlauf der Gelenke und hilft dabei, harte Übergänge oder unnötig unruhige Kurven zu erkennen.

Für kleinere Korrekturen stellt das Add-on eine Glättung auf Basis der Blender-F-Curves bereit (Abbildung 12). Änderungen können außerdem auf einer zusätzlichen NLA-Ebene liegen, ohne die importierte Grundbewegung zu überschreiben. Dieser Aufbau eignet sich für zwei Arbeitsweisen:

- Beim **Retarget-and-clean**-Workflow bleibt die GMR-Bewegung die Grundlage und wird gezielt angepasst.
- Bei einer manuellen Choreografie werden Posen und Übergänge vollständig in Blender erstellt.

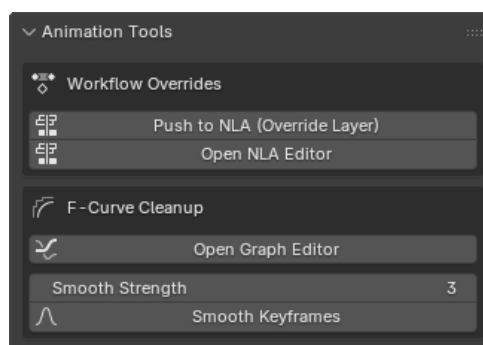


Abbildung 12: Animationsbereich des K1 Animator mit Optionen für NLA- und Graph-Editor sowie Keyframe-Glättung (eigener Screenshot)

Der K1 Animator verbindet damit die Formatkonvertierung mit einer visuellen Bearbeitungsstufe. Bewegungsdaten werden am K1-Modell sichtbar und können vor dem Training oder der Integration gezielt gestaltet werden.

4.5 Export

Der Export unterstützt die folgenden Zielformate (Tabelle 4):

Format	Ziel	Inhalt
CSV	Isaac-Werkzeuge	Root-Pose und 22 Gelenkwinkel je Frame
NPZ	Isaac Lab	vollständige Referenzbewegung für das Training
CFG (Keyframe)	B-Human	Gelenkwinkel, Dauer und Steifigkeit
CFG (Trajektorie)	B-Human	Referenzpositionen und - geschwindigkeiten für eine Policy

Tabelle 4: Exportformate des K1 Animator

Beim Export kann die Animation auf eine frei gewählte Zielbildrate neu abgetastet werden. Optional werden nur ausgewählte Gelenke oder nur der Oberkörper ausgegeben. Das ist vor allem für Keyframe-Gesten hilfreich. Beim NPZ-Export wird dagegen die vollständige Ganzkörperbewegung benötigt.

Für den Policy-Pfad ist die Zielbildrate trotz dieser freien Auswahl auf 50 Hz festgelegt (Abbildung 13). Die verwendete Isaac-Lab-Umgebung erhöht den Referenzindex einmal pro 20-ms-Policy-Schritt, und die EmoteEngine interpretiert ebenfalls jeden Trajektorienframe als 20 ms. Eine andere Exportbildrate würde den Tanz daher mit einer falschen Geschwindigkeit ausführen. Bei Keyframe-CFG-Dateien ist die Zeit dagegen explizit im Feld `duration` jedes Frames enthalten.

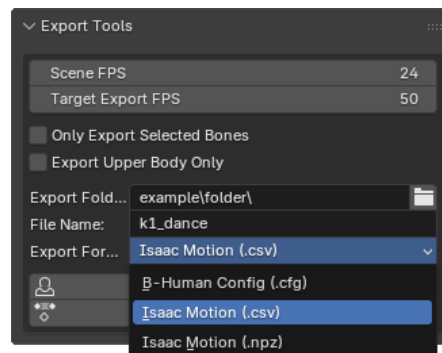


Abbildung 13: Exportbereich des K1 Animator mit den verfügbaren Zielformaten (eigener Screenshot)

4.6 Typischer Ablauf

Für den Macarena-Tanz werden das K1-Rig erzeugt, die GMR-PKL importiert und die Bewegung auf der Timeline geprüft. Aus derselben Blender-Action entstehen anschließend die NPZ-Referenz für das Training und die CFG-Referenztrajektorie für B-Human. Beim Dab endet der Ablauf dagegen mit dem direkten Export einer Keyframe-CFG.

Die Bereinigung der Bewegung im K1 Animator und der direkte Export der Keyframe-CFG werden außerdem in Video 2 zusammenhängend gezeigt.

5 Training des Ganzkörperanzes in Isaac Lab

5.1 Warum ein Training nötig ist

Die Blender-Animation beschreibt, welche Pose der K1 zu jedem Zeitpunkt einnehmen soll. Sie enthält jedoch keine Reaktion auf den tatsächlichen Zustand des Roboters. Bei einem Ganzkörperanz bewogen sich Arme, Beine und Rumpf gleichzeitig. Schon kleine Abweichungen können dazu führen, dass die folgenden Posen nicht mehr zur aktuellen Situation passen.

Für den Macarena-Tanz wird deshalb eine Tracking-Policy trainiert. Ihr Ziel ist nicht, eine neue Choreografie zu erfinden. Sie soll der vorgegebenen Referenzbewegung folgen und dabei auf den simulierten Zustand des K1 reagieren. Die Blender-Animation bleibt also der inhaltliche Tanz; das Training ergänzt eine zustandsabhängige Ausführung.

Eine Alternative wäre eine vollständig modellbasierte Regelung, die für jede Pose passende Gleichgewichts- und Kontaktbedingungen löst. Für diese Arbeit wird stattdessen die bestehende Motion-Tracking-Umgebung aus `booster_train` genutzt, weil Robotermodell, Aktuatoren und Trainingsablauf bereits auf den K1 ausgerichtet sind (BoosterRobotics, 2026b).

5.2 Trainingsumgebung

Das Training findet in Isaac Lab statt, einem auf Isaac Sim aufbauenden Framework für robotisches Lernen (Mittal et al., 2025). Die Umgebung simuliert den K1 auf Grundlage seines URDF-Modells und lädt die aus Blender exportierte NPZ-Datei als Referenzbewegung. Viele Roboterinstanzen laufen parallel und sammeln unterschiedliche Erfahrungen für dasselbe Bewegungsziel.

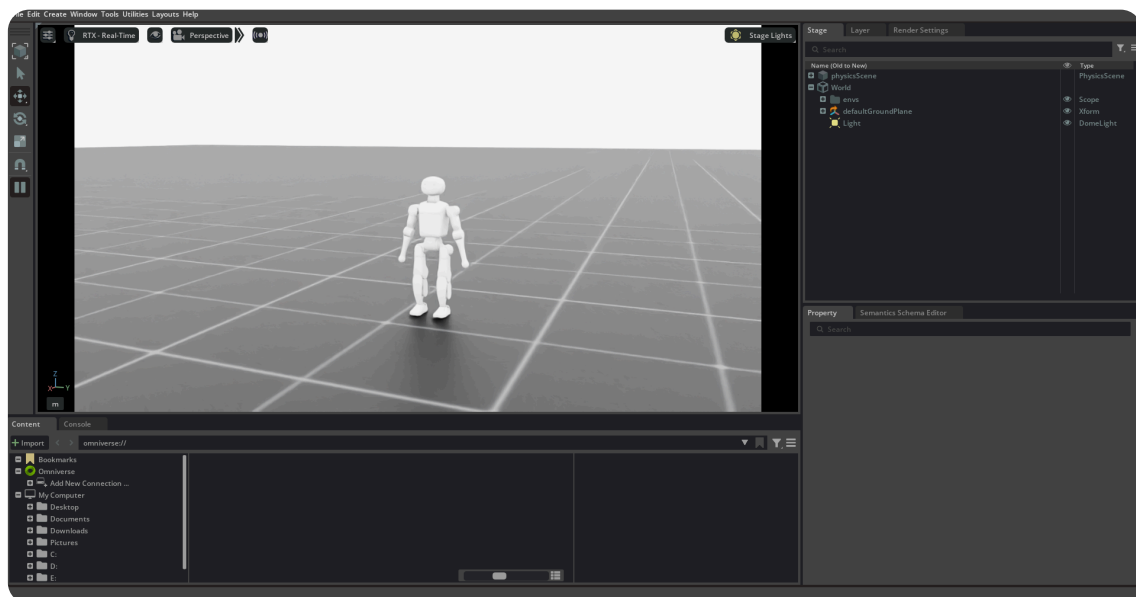


Abbildung 14: Booster K1 im Isaac-Sim-Viewport, der Physik-Engine hinter Isaac Lab (eigener Screenshot)

Die Physik wird in 5-ms-Schritten mit 200 Hz berechnet. Nach jeweils vier Physikschritten entsteht eine neue Aktion, sodass die Policy mit 50 Hz arbeitet. Dieselbe 20-ms-Zeitbasis wird auch in B-Human verwendet.

Für das Training sind 4096 parallele Umgebungen konfiguriert. Der überwiegende Teil des Terrains ist nahezu flach, ein kleinerer Teil enthält stärkere Unebenheiten. Zusätzlich variiert die Umgebung unter anderem Reibungs- und Rückpralleigenschaften der Kontaktflächen, kleine Gelenk-Nullpunktfehler, den Schwerpunkt des Rumpfes und die Verzögerung der Aktuatoren. In zufälligen Abständen erhält der Roboter kurze Störungen. Solche Variationen werden beim simulativen Training eingesetzt, damit eine Policy nicht ausschließlich an eine einzelne Parameterbelegung angepasst wird (Peng, Andrychowicz, et al., 2018).

Die Macarena-Konfiguration setzt für Beine, Füße, Arme und Kopf feste PD-Parameter. Sie bestimmen gemeinsam mit den Aktionsskalierungen, wie die von der Policy ausgegebenen Werte in Gelenkziele umgesetzt werden. Für den dokumentierten Trainingslauf wurden die in Tabelle 5 zusammengefassten Werte verwendet. In B-Human wird dafür während des Tanzes ein eigenes Motorprofil gewählt.

Gelenkgruppe	Steifigkeit	Dämpfung	Aktionsskalierung
Hüfte und Knie	80	4	0,0625–0,125
Fußgelenke	50 beziehungsweise 60	2	0,1667
Arme	4	1	0,875
Kopf	4	1	0,375

Tabelle 5: Aktuator- und Aktionsparameter des verwendeten Macarena-Trainingslaufs

5.3 Beobachtung und Aktion

In jedem Policy-Schritt werden Referenz und aktueller Roboterzustand zusammengeführt. Der Eingang besteht aus fünf Gruppen:

Gruppe	Werte	Bedeutung
Referenzbewegung	44	gewünschte Gelenkpositionen und -geschwindigkeiten
Rumpforientierung	6	relative Ausrichtung zur Referenz
Drehrate des Rumpfes	3	gemessene Winkelgeschwindigkeit
aktueller Gelenkzustand	44	Positionen und Geschwindigkeiten der Gelenke
vorherige Aktion	22	Ausgabe des letzten Policy-Schritts
Summe	119	

Tabelle 6: Zusammensetzung der Policy-Eingabe

Die Gelenkgeschwindigkeiten bilden einen Teil des aktuellen Gelenkzustands und werden im Training durch einen exponentiellen Tiefpass geglättet. In B-Human stammen die

entsprechenden 22 Werte aus der Roboteranbindung beziehungsweise der Simulation und werden in derselben Gelenkreihenfolge in die Beobachtung eingetragen.

Die Policy gibt 22 Werte aus, einen für jedes Gelenk. Diese Werte werden nicht direkt als Winkel verwendet, sondern mit gelenkspezifischen Faktoren skaliert und zur jeweiligen Grundstellung addiert. Dadurch kann derselbe Netzausgang an die unterschiedlichen Bewegungsbereiche von Beinen, Armen und Kopf angepasst werden.

Die Reihenfolge und Bedeutung dieser Werte sind Teil der Schnittstelle zwischen Training und B-Human. Die EmoteEngine stellt dieselben Beobachtungsgruppen in dieser Reihenfolge bereit und bildet die 22 Ausgaben mit denselben Skalierungen auf die K1-Gelenke ab. Kapitel 6 greift diese Zuordnung auf der Laufzeitseite erneut auf, ohne den gesamten Trainingsaufbau zu wiederholen.

5.4 Lernziel

Das Training verwendet PPO (Schulman et al., 2017). Actor und Critic besitzen jeweils drei Hidden-Layer mit 512, 256 und 128 Einheiten und verwenden ELU-Aktivierungen. Der Actor berechnet die 22 Gelenkaktionen, während der Critic nur während des Trainings bei der Bewertung der Zustände hilft. Eine PPO-Aktualisierung verarbeitet jeweils 24 Schritte pro Umgebung in fünf Epochen mit jeweils vier Mini-Batches. Verwendet werden eine anfängliche Lernrate von 10^{-3} mit adaptiver Anpassung, $\gamma = 0,99$, $\lambda = 0,95$ und ein Entropiekoeffizient von 0,005.

Die Belohnungsfunktion verbindet mehrere Ziele. Positionen, Orientierungen und Geschwindigkeiten ausgewählter Körperteile sollen zur Referenz passen. Füße, Hände und Rumpf erhalten zusätzliche Tracking-Terme, damit ihre für den Tanz besonders wichtigen Posen stärker gewichtet werden. Zusätzlich werden starke Änderungen der Aktionen, Gelenkgrenzverletzungen und unerwünschte Bodenkontakte bestraft. Tabelle 7 fasst die Gewichte des verwendeten Laufs zusammen.

Reward-Gruppe	Gewicht	Bedeutung
globaler Rumpfanker	0,5 / 0,5	Position / Orientierung
alle verfolgten Körper	je 1,0	Position, Orientierung sowie lineare und angulare Geschwindigkeit
Füße	1,5 / 1,5	Position / Orientierung
Hände	1,0 / 1,0	Position / Orientierung
Rumpf	2,0 / 1,0	Position / Orientierung
Aktionsänderung, Gelenkgrenze, Fremdkontakt	je $-1,0$	Strafterme

Tabelle 7: Reward-Gruppen und Gewichte des Macarena-Trainings

Eine Episode endet nach zehn Sekunden oder vorzeitig, wenn die vertikale Abweichung des Rumpfankers beziehungsweise eines Fußes oder einer Hand 0,25 m überschreitet. Zusätzlich wird bei einer zu großen Abweichung der Rumpfneigung abgebrochen. Horizontale Verschiebungen lösen für sich genommen keinen Abbruch aus.

5.5 Trainingsablauf

Eine Episode kann an unterschiedlichen Stellen des Tanzes beginnen. Bereiche, in denen die Policy häufiger scheitert, werden beim erneuten Start häufiger ausgewählt. Dadurch verbringt das Training mehr Zeit mit schwierigen Teilen der Bewegung, ohne die übrigen Frames vollständig auszublenden.

Der Ablauf wiederholt sich in allen Umgebungen:

1. Referenzframe und aktueller Zustand werden zur Beobachtung zusammengesetzt.
2. Der Actor erzeugt neue Gelenkaktionen.
3. Die Simulation führt die Aktion über mehrere Physikschrte aus.
4. Belohnungen und Abbruchbedingungen bewerten den neuen Zustand.
5. PPO aktualisiert Actor und Critic anhand der gesammelten Erfahrungen.

Die Konfiguration erlaubt bis zu 20.000 PPO-Iterationen. Der hier ausgewertete Lauf wurde vorher manuell beendet. Für die Dokumentation wurden die dabei gespeicherte, vollständig aufgelöste Umgebungs- und Agentenkonfiguration sowie die TensorBoard-Daten ausgewertet.

5.6 Trainingsverlauf

Die mittlere Episodenbelohnung und die mittlere Episodenlänge zeigen zwei verschiedene Seiten des Lernverlaufs. In den folgenden Diagrammen stellt die dünne Linie die protokollierten Mittelwerte dar. Die dicke Linie zeigt einen exponentiell geglätteten Trend, an dem die längerfristige Entwicklung besser erkennbar ist.



Abbildung 15: Mittlere Episodenbelohnung bis Trainingsiteration 11.026 (eigene Darstellung auf Grundlage der TensorBoard-Trainingsdaten)

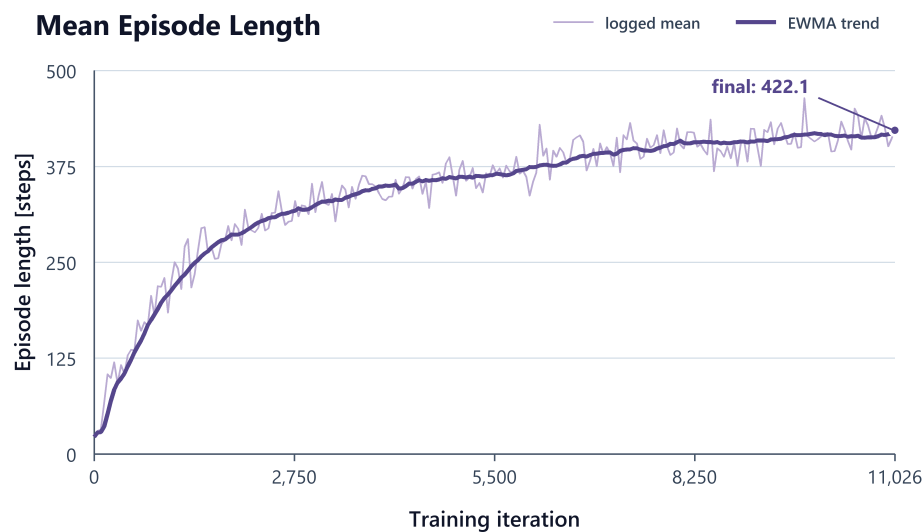


Abbildung 16: Mittlere Episodenlänge bis Trainingsiteration 11.026 (eigene Darstellung auf Grundlage der TensorBoard-Trainingsdaten)

Die mittlere Episodenbelohnung steigt schnell aus dem negativen Bereich und erreicht am Ende 62,8. Der später abgeflachte Trend zeigt nur noch kleinere Verbesserungen des gemeinsamen Lernziels. Der Wert ist keine Prozentangabe, sondern fasst Referenzverfolgung und Strafterme zusammen.

Die mittlere Episodenlänge wächst ebenfalls schnell, nähert sich einem Plateau und endet bei 422,1 von höchstens 500 Schritten. Die Policy löst damit seltener einen vorzeitigen Abbruch aus. Die Episodenlänge allein belegt jedoch weder die Genauigkeit jeder Pose noch die Stabilität auf dem realen Roboter.

Der Lauf wurde nach ungefähr 16 Stunden bei der protokollierten Trainingsiteration 11.026 manuell beendet. Das Verhalten wurde als ausreichend eingeschätzt und beide Kurven verbesserten sich nur noch langsam; ein festes Konvergenzkriterium wurde nicht verwendet.

BEGLEITVIDEO

VIDEO 3 - LERNFORTSCHRITT DER MACARENA-POLICY

Link: youtu.be/RT7jXDQAa-U



5.7 Ergebnis und Export

Das exportierte ONNX-Modell ist das finale Ergebnis für die Laufzeit. Es enthält den Actor mit integrierter Beobachtungsnormalisierung, aber weder Critic noch PPO-Trainingszustand. Die EmoteEngine konfiguriert seine 119 Eingabe- und 22 Ausgabewerte zusammen mit der CFG-Referenztrajektorie derselben Blender-Action.

6 Integration in B-Human

6.1 Übergabe aus dem Training

Für einen Policy-Tanz benötigt B-Human zwei Dateien: das trainierte ONNX-Modell und die zugehörige CFG-Referenztrajektorie. Das Modell berechnet die Gelenkaktionen. Die Trajektorie gibt vor, welcher Teil des Tanzes zum aktuellen Zeitpunkt verfolgt werden soll.

Aus derselben Blender-Action erzeugt der K1 Animator zwei auf ihren jeweiligen Zweck zugeschnittene Exporte: Die NPZ-Datei dient als Referenz für das Training in Isaac Lab, während B-Human die CFG-Referenztrajektorie zusammen mit dem trainierten ONNX-Modell verwendet. Die Länge des Tanzes ergibt sich aus der Anzahl der Referenzframes und der festen Schrittzeit; sie muss nicht zusätzlich von Hand eingetragen werden. Die Schnittstellenübersicht im Anhang 14 ordnet die Dateien beider Beispielpfade ihren Erzeugern und Verbrauchern zu.



Abbildung 17: Ausführung des Macarena-Tanzes durch die EmoteEngine in SimRobot
(eigener Screenshot)

6.2 Architektur der Ausführung

Die Integration folgt dem vorhandenen Aufbau von B-Human. Eine Entscheidung im Verhaltenssystem führt nicht direkt zu Gelenkwinkeln, sondern durchläuft drei Ebenen:

Ebene	Entscheidung	Ergebnis
Strategy	Soll ein Jubel stattfinden?	Anforderung mit Jubelrolle und Auswahl
Skill	Welche Bewegung wird angefordert?	regulärer Emote-MotionRequest
Motion	Wie wird die Bewegung ausgeführt?	Gelenkziele aus Policy oder Keyframes

Tabelle 8: Datenfluss vom Spielereignis bis zur Bewegung

Der Bewegungswunsch enthält die Jubelrolle und optional die Auswahl eines bestimmten Emotes. Die Rolle unterscheidet zwischen Torschütze und unterstützenden Robotern. Ob der ausgewählte Eintrag als Policy oder als Keyframe-Animation ausgeführt wird, entscheidet erst die Konfiguration der EmoteEngine. Damit bleibt die Verhaltensseite unabhängig von der technischen Umsetzung und den konkreten Modell- und Dateinamen.

Im Motion-Thread erzeugt die EmoteEngine die eigentliche Bewegung. Sie setzt den aktuellen Referenzframe, den gemessenen Roboterzustand und die vorherige Aktion zur Policy-Eingabe zusammen. Alle 20 ms wird das ONNX-Modell ausgewertet. Die daraus entstehenden 22 Aktionen werden auf die Gelenke des K1 abgebildet.

Für die Gelenkgeschwindigkeiten verwendet die EmoteEngine `JointAngles.velocity` aus der Roboteranbindung beziehungsweise der Simulation. Beim realen K1 wird dieser Wert aus `dq` im Motorzustand übernommen. Damit steht der Policy in beiden Laufzeitumgebungen dieselbe Art von Zustandsinformation zur Verfügung.

Der Motion-Thread selbst läuft schneller als die Policy. Zwischen zwei Policy-Schritten bleiben die zuletzt berechneten Gelenkziele aktiv. Diese Trennung erlaubt es, das Modell mit derselben Frequenz wie im Training auszuführen, ohne den allgemeinen Robotertakt zu verändern.

6.3 EmoteEngine

Die EmoteEngine ist die Laufzeitkomponente für Torjubel. Sie verwaltet die geladenen Bewegungen und erzeugt die passende Motion-Phase. Jeder Eintrag besitzt einen Typ: Policy-Emotes bestehen aus Modell und Trajektorie, Keyframe-Emotes verweisen auf eine Animationsdatei. Die Einträge sind in zwei rollenbezogene Listen für Torschützen und Unterstützer gegliedert. Beide Listen dürfen beide Ausführungsarten enthalten. Ein neuer Jubel kann dadurch einer Rolle zugeordnet werden, ohne die Engine oder die Verhaltenslogik zu ändern.

Ein Emote kann über einen festen Index gewählt werden. Alternativ wird ein Standardwert oder eine zufällige Auswahl unter den erfolgreich geladenen Einträgen verwendet. Die einmal getroffene Auswahl bleibt während derselben Torfeier erhalten. Optional kann

die Zufallsauswahl über einen gemeinsamen Startwert im Team synchronisiert werden. Für die Tänze dieser Arbeit wählen die Roboter unabhängig voneinander.

Ausführungsart	Konfiguration	gesteuerte Bewegung
Policy	ONNX + Referenz-CFG	vollständiger Körper
Keyframe	Animations-CFG	Kopf und Arme; Beine im Stand

Tabelle 9: Die beiden Ausführungsarten der EmoteEngine

Die Konfiguration dieser Arbeit ordnet dem Torschützen die Policy-Tänze *macarena* und *mj* zu. Unterstützende Roboter wählen zwischen den sechs Keyframe-Emotes *clap*, *dab*, *wave*, *sweep*, *groove* und *sixseven*. Diese Zuordnung ist eine Konfigurationsentscheidung und keine feste Kopplung zwischen Rolle und Ausführungsart.

Video 4 zeigt alle acht konfigurierten Emotes nacheinander in SimRobot.

BEGLEITVIDEO

VIDEO 4 - B-HUMAN-EMOTES IM ÜBERBLICK

Link: youtu.be/9T_RV-ecGfE



6.4 Gemeinsame Laufzeiteinstellungen

Die gemeinsame *EmoteSettings*-Repräsentation legt zentral fest, ob Torjubel aktiviert sind und wie lange regulär angeforderte Emotes höchstens abgespielt werden. Im verwendeten Stand sind die Torjubel aktiviert; die maximale aktive Wiedergabedauer beträgt zehn Sekunden.

Bei der automatischen Torfeier beendet die Strategy die Anforderung nach zehn Sekunden; unabhängig davon beginnt auch die EmoteEngine für reguläre Anforderungen spätestens dann den Übergang zum Stand. Die Motion-Phase endet erst nach dem anschließenden konfigurierten Übergang, damit die Gelenke nicht abrupt in die Standpose springen. Eine dauerhaft anliegende Anforderung oder ein als Schleife konfiguriertes Emote startet danach nicht unmittelbar neu. Der Controller-Pfad ist davon bewusst ausgenommen: Er dient dem beaufsichtigten Test, und seine Anforderung bleibt aktiv, solange die zugehörige Taste gehalten wird.

6.5 Motorparameter während des Tanzes

Policy-Emotes verwenden das Motorprofil *emote*. Die Auswahl betrifft die Reglerparameter der Gelenke, nicht die Struktur oder Ausgabe des neuronalen Netzes. Für gehaltene Tanzposen verwendet dieses Profil weichere Armparameter als das reguläre Bewegungsprofil.

Beim Keyframe-Pfad stammen die Beinziele und das Motorprofil dagegen aus dem StandGenerator. Die Animationsdatei ergänzt nur die Winkel und den Steifigkeitswert für Kopf und Arme. Dadurch behalten die Beine während einer Keyframe-Geste die für den Stand vorgesehenen Motorparameter.

SimRobot und der reale K1 lesen Verstärkung, Dämpfung und maximales Drehmoment aus derselben MotorParameters-Konfiguration. Das Emote-Profil wird damit an einer Stelle gepflegt und in beiden Laufzeitumgebungen über denselben Mechanismus ausgewählt. Es bleibt dennoch eine eigene Laufzeitkonfiguration: Die für einen sicheren Robotereinsatz gewählten Werte müssen nicht jeden Trainingsparameter unverändert übernehmen.

6.6 Wiederholung und Übergänge

Anfang und Ende eines Emotes besitzen nicht automatisch dieselbe Pose wie die vorausgehende oder folgende Bewegung. Die Übergänge müssen deshalb Teil der Ausführung sein und dürfen nicht als direkter Sprung zwischen zwei Gelenkstellungen erfolgen.

Wiederholung und Übergangszeit werden für jedes Emote einzeln konfiguriert. Die eingesetzten Emotes werden jeweils einmal abgespielt. Danach hält die Motion-Phase zunächst die Standbewegung und meldet ihren Abschluss an das Verhaltenssystem, das die Torfeier beendet und zur normalen Bewegung zurückkehrt. Für reguläre Anforderungen begrenzt zusätzlich die gemeinsame Laufzeit aus `EmoteSettings` den Emote-Ablauf.

Der Übergang beginnt in der gemessenen Roboterpose. Die EmoteEngine verwendet den StandGenerator als Ziel beziehungsweise als Grundbelegung und interpoliert zu Beginn und am Ende die von ihr kontrollierten Gelenke. Bei den Policy-Emotes werden sämtliche kontrollierten Gelenke über die konfigurierten 400 ms zur Policy beziehungsweise zurück zum Stand überblendet. Bei einer Keyframe-Animation werden nur Kopf und Arme über 400 ms eingeblendet; die Beine bleiben durchgehend beim StandGenerator. Dadurch werden direkte Sollwertsprünge vermieden.

Die Übergänge wurden zusammen mit den Emotes auf dem realen K1 ausgeführt. Dabei gelangen sowohl der Einstieg in die Bewegung als auch die anschließende Rückkehr zum Stand ohne auffällige Abweichungen gegenüber der zuvor geprüften Simulation. Kapitel 7 ordnet diesen Hardwaretest ein.

6.7 Keyframe-Pfad

Der Keyframe-Pfad ist für einfache Gesten gedacht. Kopf und Arme folgen einer zeitlich definierten Folge von Posen, während der StandGenerator weiterhin die Beine kontrolliert. Dadurch kann ein weiterer Jubel ohne Policy-Training ergänzt werden.

Eine Keyframe-Datei enthält für jeden Frame die Gelenkwinkel, die Dauer und einen Steifigkeitswert. Der K1 Animator kann eine Blender-Animation direkt in dieses CFG-Format exportieren. Während der Wiedergabe übernimmt die EmoteEngine daraus Kopf und Arme; eine interne Stand-Phase führt unabhängig davon die Beine. Die Felder des Formats und die sechs eingebundenen Keyframe-Emotes sind in Tabelle 12 und Tabelle 13 dokumentiert.

Der Vorteil dieses Pfads liegt in seiner Einfachheit. Für eine kurze Armbewegung wäre ein vollständiges RL-Training unverhältnismäßig. Bei einem dynamischen Tanz, der den gesamten Körper nutzt, bietet die Policy dagegen die passendere Schnittstelle.

6.8 PuppeteerEngine

Die PuppeteerEngine ist ein zusätzliches Werkzeug in SimRobot und kein Bestandteil der Laufzeit im realen Spiel. Im Puppeteer-Modus bleibt die Physiksimulation aktiv, während die Schwerkraft deaktiviert und die Position des Roboters fixiert wird. Der K1 kann über Gelenkregler direkt in eine Pose gebracht werden. Außerdem lassen sich vorhandene Keyframe-CFG-Dateien mit einstellbarer Dauer, Steifigkeit, Schleifenoption und Geschwindigkeit abspielen.

Die geladenen CFG-Dateien verwenden dieselbe Datenstruktur, die auch die EmoteEngine für Keyframe-Emotes liest. Blender und die PuppeteerEngine erfüllen dabei unterschiedliche Aufgaben. Der K1 Animator übernimmt Import, Kurvenbearbeitung, IK und den Export der Keyframe-CFG. Die PuppeteerEngine dient dazu, einzelne Posen manuell zu prüfen und die exportierten Sequenzen in SimRobot abzuspielen. Beide Abspieler verwenden dieselbe Zeitsemantik: Die Dauer eines Frames beschreibt die Interpolation von diesem Frame zum folgenden. Der Anhang B dokumentiert das Format im Detail.

6.9 Auslösung des Torjubels

Ein Emote kann auf drei Wegen gestartet werden:

- automatisch nach einem erkannten eigenen Tor,
- manuell über Testbefehle,
- über einen angeschlossenen Controller.

Für die automatische Auslösung beobachtet die Strategy den Spielstand. Nach einem eigenen Tor wird der wahrscheinliche Torschütze aus den zuletzt ballnahen Spielern bestimmt. Dieser Roboter erhält die Torschützenrolle, die übrigen aktiven Teammitglieder die Unterstützerrolle. Erst die jeweilige Rollenliste legt fest, welcher Tanz und welcher Ausführungstyp verwendet wird. Liegt kein geeigneter jüngerer Ballkontakt vor, wird zunächst die ballspielende Rolle und danach der eigene Roboter als Fallback verwendet.

Da nach einem Tor nicht immer eindeutig feststeht, welcher Roboter zuletzt entscheidend am Ball war, wird der Torschütze aus den zuletzt ballnahen Spielern geschätzt. Diese Zuordnung bestimmt ausschließlich, welcher Roboter ein Torschützen-Emote und welche Roboter ein Unterstützer-Emote erhalten. Sie verändert keine spielentscheidende Verhaltensentscheidung. Aktivierung und Zeitgrenze gelten gemeinsam für automatische und reguläre manuelle Anforderungen. Für den beaufsichtigten Controller-Test gilt diese Zeitgrenze nicht, da die Vorführung jederzeit manuell beendet werden kann.

Für Tests und Vorführungen genügt es, das Steuerkreuz nach unten zu halten: Die Joystick-Option fordert dann die konfigurierte Emote-Rolle und den konfigurierten Index an; beim Loslassen endet die Anforderung und die EmoteEngine blendet zum Stand zurück. Im verwendeten Stand ist dafür die Torschützenrolle ausgewählt. Der Controller-Pfad läuft über Skill und Motion, aber nicht über die automatische Torerkennung der Strategy.

Video 5 zeigt eine automatisch ausgelöste Torfeier in einem simulierten Spiel in SimRobot: Der Torschütze führt zufällig einen der beiden Policy-Tänze aus, die Unterstützer jeweils zufällig eines der Keyframe-Emotes.

BEGLEITVIDEO

VIDEO 5 - TORJUBEL IM SPIEL

Link: youtu.be/p9NdwmyqxGg



6.10 Einordnung in die Sicherheitsstruktur

Ein Emote umgeht den normalen B-Human-Optionsbaum nicht. Höher priorisierte Zustände, insbesondere der Stoppzustand, können den Jubel deshalb weiterhin unterbrechen. Ist die Funktion in `EmoteSettings` deaktiviert, werden keine Assets geladen und jede Anforderung führt in den Stand. Bei aktivierter Funktion werden die konfigurierten Assets einmalig geladen. Kann ein Modell oder eine Animation nicht geladen werden, fällt die Bewegung ebenfalls auf die Standbewegung zurück. Auch eine erzwungene Sitzbewegung beendet den Emote-Pfad; die EmoteEngine kehrt dann in den Stand zurück.

Diese Einordnung ist wichtiger als eine eigene, parallele Sicherheitslogik innerhalb des Tanzes. Die EmoteEngine erzeugt eine neue Motion, bleibt aber Teil der bestehenden Robotersteuerung.

6.11 Zusammenspiel der Komponenten

Die Integration führt Policy-Tänze und Keyframe-Emotes in einem gemeinsamen Ablauf zusammen. Der K1 Animator erzeugt die benötigten Bewegungsdateien, die EmoteEngine führt sie innerhalb der B-Human-Motion-Architektur aus und das Verhaltenssystem wählt Rolle und Auslösezeitpunkt. Die PuppeteerEngine ergänzt die Pipeline als Prüf- und Abspielwerkzeug für exportierte Keyframe-Bewegungen.

7 Evaluation und Diskussion

7.1 Ziel der Evaluation

Die Evaluation prüft, ob die entwickelte Pipeline ihre drei Kernaufgaben erfüllt:

1. Eine menschliche Bewegung muss auf den K1 übertragen, im K1 Animator am Robotermodell bearbeitet und in den für die weiteren Schritte benötigten Formaten ausgegeben werden können.
2. Dynamische Ganzkörperbewegungen müssen über den Policy-Pfad und einfachere Gesten über den direkten Keyframe-Pfad ausführbar sein. Dabei müssen die jeweiligen Schnittstellen zu Isaac Lab und B-Human zusammenpassen.
3. Beide Ausführungsarten müssen über die reguläre B-Human-Architektur ausgelöst und in SimRobot sowie abschließend auf dem realen K1 geprüft werden.

Damit orientiert sich die Prüfung direkt an der Forschungsfrage. Bewertet wird nicht nur ein einzelner Tanzframe, sondern der Zusammenhang zwischen den Stufen.

7.2 Prüfung des Bearbeitungs- und Exportablaufs

Für den Macarena-Tanz wurde die von GMR erzeugte PKL-Datei in den K1 Animator importiert und als Animation auf dem K1-Rig geprüft. Die Bewegung ließ sich auf der Timeline abspielen, an einzelnen Gelenken bearbeiten und in die für Training und Laufzeit vorgesehenen Formate ausgeben. Der Dab wurde dagegen als kurze Keyframe-Animation bearbeitet und ohne Policy-Training als Animations-CFG exportiert.

Die beiden Beispiele decken damit die in dieser Arbeit benötigten Exportpfade ab: eine Ganzkörperreferenz für das Policy-Training, eine Referenztrajektorie für die Policy-Ausführung und eine eigenständige Keyframe-Animation. Diese funktionale Prüfung bewertet gezielt den tatsächlich verwendeten Datenfluss dieser beiden Beispiele.

7.3 Schnittstelle zwischen Training und B-Human

Die Trainingskonfiguration und die EmoteEngine wurden entlang ihres Datenflusses abgeglichen. Beide Seiten verwenden dieselben Gruppen von Eingaben: Referenzbewegung, relative Rumpforientierung, Drehrate, Gelenkzustand und vorherige Aktion. Reihenfolge, Gelenkzuordnung und Grundstellungen sind festgelegt. Die Aktionsskalierungen werden aus der verwendeten Trainingskonfiguration übernommen und gehören zusammen mit dem finalen Modell zur Laufzeitschnittstelle.

Das ONNX-Modell enthält die während des Trainings gelernte Beobachtungsnormalisierung. Die EmoteEngine muss deshalb keine eigenen Mittelwerte verwalten. Die Policy wird in beiden Systemen mit derselben Schrittzeit ausgeführt.

Prüfpunkt	Vergleich	Ergebnis
Policy-Eingabe	119 Werte, Aufbau und Reihenfolge	code-seitig abgeglichen
Policy-Ausgabe	22 Gelenkaktionen	Dimensionen stimmen überein
Gelenkmapping	Zuordnung und Skalierung	code-seitig abgeglichen
Zeitbasis	Isaac Lab und EmoteEngine	jeweils 20 ms

Tabelle 10: Geprüfte Schnittstellen zwischen Training und Laufzeit

Die Prüfung vergleicht die aufgelöste Trainingskonfiguration, die Ein- und Ausgabedimensionen des ONNX-Modells und den Quellcode der EmoteEngine. Sie belegt damit Aufbau und Zuordnung der Schnittstelle, nicht die numerische Gleichheit aufgezeichneter Sensorwerte. Insbesondere werden die Gelenkgeschwindigkeiten im Training geglättet, während B-Human die von Roboteranbindung beziehungsweise Simulation bereitgestellten Werte verwendet. Diese Abgrenzung wird bereits in Kapitel 5 beschrieben.

7.4 Funktionale Prüfung in der Simulation

Der Policy-Tanz wurde während des Trainings in Isaac Sim ausgeführt. Anschließend wurden ONNX-Modell und Referenztrajektorie in B-Human eingebunden und der Macarena-Tanz in SimRobot gestartet. Damit durchläuft die Bewegung die vorgesehenen Stufen vom Blender-Export bis zur EmoteEngine.

Zusätzlich wurden die beiden Ausführungsarten, die rollenbezogene Auswahl und die Auslösung über den regulären Behavior-Pfad integriert. Die Konfiguration dieser Arbeit ordnet die Policy-Tänze dem Torschützen und die Keyframe-Emotes den Unterstützern zu. Bei regulären Anforderungen beendet die gemeinsam ausgewertete Zeitgrenze auch dauerhaft angeforderte oder wiederholte Emotes und leitet den Übergang zum Stand ein. Der beaufsichtigte Controller-Test kann diese Grenze gezielt übergehen.

Anforderung	Stand	Nachweis
K1-Bewegung in Blender bearbeiten	umgesetzt	Import, Rig, Timeline und Export
Ganzkörper Tanz trainieren	umgesetzt	Isaac-Lab-Lauf und ONNX-Export
Policy in B-Human ausführen	umgesetzt	Macarena in SimRobot
Keyframe-Tanz ohne Training	umgesetzt	Dab mit StandGenerator
Jubel über Behavior auslösen	umgesetzt	Strategy-Skill-Motion-Pfad
Ausführung auf dem realen K1	umgesetzt	aufgezeichnete Hardwaretests

Tabelle 11: Funktionale Anforderungen und ihre Nachweise

7.5 Emotes auf dem realen Roboter

Die abschließende Hardwareprüfung ergänzte die Simulation um die tatsächliche Gelenk- und Reglerdynamik des K1. Dabei wurden sowohl Policy- als auch Keyframe-Emotes auf dem realen Roboter ausgeführt und aufgezeichnet. Die Bewegungen ließen sich wie vorgesehen starten und beenden. Auch die Übergänge aus dem Stand und zurück zum Stand verliefen ohne auffällige Sprünge.

BEGLEITVIDEO

VIDEO 6 - EMOTES AUF DEM ECHTEN ROBOTER

Link: youtu.be/6btjwP2nQYM



Im sichtbaren Bewegungsablauf zeigten die Hardwaretests keine wesentlichen Abweichungen gegenüber den zuvor in Isaac Sim und SimRobot geprüften Ausführungen. Damit ist der vollständige technische Pfad vom Bewegungsentwurf bis zur Ausführung auf dem realen K1 für die Beispiele dieser Arbeit funktional nachgewiesen. Der Test ersetzt keine längerfristige Belastungs- oder Spielsituationsprüfung; er zeigt, dass die implementierten Schnittstellen und Bewegungen auf der Zielhardware zusammenarbeiten.

7.6 Beantwortung der Forschungsfrage

Die umgesetzte Pipeline bildet einen zusammenhängenden Weg von der Bewegungsvorlage bis zum Torjubil in B-Human. Die menschliche Bewegung kann auf den K1 übertragen und im K1 Animator am Robotermodell bearbeitet werden. Aus derselben Arbeitsumgebung entstehen mit dem Macarena-Tanz die Dateien für einen trainierten Ganzkörperanz und mit dem Dab eine direkte Keyframe-Animation.

Die B-Human-Integration trennt weiterhin zwischen Verhaltensentscheidung und Bewegungsausführung. Dadurch kann der Jubel automatisch oder manuell angefordert werden, ohne eine zweite, unabhängige Steuerungsarchitektur einzuführen. Beide Ausführungspfade wurden zunächst in SimRobot und anschließend auf dem realen K1 innerhalb dieses Ablaufs geprüft. Die sichtbaren Hardwareergebnisse entsprachen dabei der zuvor beobachteten Simulation.

Der K1 Animator ist dabei die gemeinsame Bearbeitungsoberfläche der entwickelten Pipeline. Er macht Bewegungen am K1-Modell zugänglich, unabhängig davon, ob sie aus Motion Capture stammen oder manuell erstellt wurden. Das Policy-Training und die B-Human-Integration verwenden anschließend die jeweils für ihren Zweck exportierten Repräsentationen.

7.7 Grenzen der Evaluation

Die Evaluation konzentriert sich auf zwei konkrete Bewegungspfade und prüft den technischen Ablauf. Sie umfasst weder eine Nutzerstudie zur Bedienbarkeit des K1 Animator noch einen Vergleich mit anderen Animationswerkzeugen. Aussagen zur visuellen Qualität und zur Übertragung auf die Hardware bleiben deshalb auf die gezeigten Beispiele und die durchgeführten Testläufe beschränkt. Eine längerfristige Ausführung unter den Bedingungen eines realen Spiels wurde nicht untersucht.

Die Schnittstelle zwischen Training und B-Human wurde anhand von Konfiguration, Metadaten und Quellcode geprüft. EmoteEngine und PuppeteerEngine laden dieselbe Keyframe-Struktur und ordnen die Dauer eines Frames jeweils dem Übergang zu dessen folgendem Frame zu. Dieser Abgleich betrifft die Quellcode- und Formatschnittstelle; die PuppeteerEngine bleibt ein Prüfwerkzeug für die Simulation und ist nicht Bestandteil der Laufzeit auf dem realen Roboter.

Eine weitere praktische Grenze ist der Speicherbedarf der Bewegungsdateien. Die größere der beiden eingesetzten Policy-Referenztrajektorien belegt rund 754 KiB, die zweite rund 356 KiB. Die sechs Keyframe-Dateien sind mit jeweils etwa 27 bis 83 KiB deutlich kleiner, umfassen zusammen jedoch ebenfalls rund 338 KiB. Hinzu kommen bei Policy-Emotes die jeweiligen ONNX-Modelle. Bei aktivierter Funktion werden die konfigurierten Assets einmalig geladen, weshalb nicht nur der Speicher im Repository, sondern auch die zur Laufzeit geladene Auswahl relevant ist. Für die begrenzte Zahl der Beispiele ist dieser Umfang handhabbar; vor einer größeren Auswahl aktiv geladener Emotes im Spielbetrieb sollten Datenreduktion, komprimierte Speicherung und eine geeignete Interpolation zwischen weniger Referenzframes untersucht werden.

7.8 Grenzen des Retargetings

GMR bildet nicht jede menschliche Pose fehlerfrei auf den K1 ab. An kinematisch schwer erreichbaren Stellen springt insbesondere die Schulter gelegentlich abrupt in eine stark abweichende Rotation, wenn sich die Zielpose sonst nicht erreichen lässt und die Schulter stattdessen einmal schnell in die entgegengesetzte Richtung dreht, um dorthin zu gelangen.

Auch der Bodenkontakt der Füße ist betroffen: GVHMR verliert dabei teils die Information, ob ein Fuß tatsächlich leicht angehoben war, sodass das Training in solchen Momenten weiterhin von Bodenkontakt ausgeht. Zusammen mit der sichtbar ungenauen rekonstruierten Fußrotation kann das zu Fehlern im Training führen.

Beim Retargeting von GVHMR zu GMR gehen außerdem die Handinformationen verloren, da der K1 keine Hände besitzt. Zusammen mit dem weitgehend starren Torso fehlen damit Elemente, die für einen natürlichen Tanz wichtig sind: Hüftbewegung und Wirbelsäule lassen sich kaum nachbilden, was sich beim Macarena-Tanz am deutlichsten am abschließenden Hüftschwung zeigt, der im Vergleich zum Originalvideo merklich schwächer ausfällt.

Eine weitere praktische Einschränkung betrifft den NPZ-Export des K1 Animator für das Policy-Training. Er bildet das Ergebnis des von `booster_train` bereitgestellten CSV-zu-NPZ-Skripts nicht bitgenau nach. Visuell ist der Unterschied kaum wahrnehmbar. Im Training wirken sich aber bereits kleine numerische Abweichungen aus. Für das Training wird deshalb empfohlen, stattdessen die CSV-Datei zu exportieren und mit dem offiziellen Skript von `booster_train` in eine NPZ-Datei umzuwandeln.

Trotz dieser Einschränkungen ist das Bewegungsergebnis für den Zweck dieser Arbeit bereits geeignet. Auf dem realen K1 ist der Tanz visuell kaum von der zuvor geprüften Simulation zu unterscheiden.

8 Fazit und Ausblick

8.1 Fazit

Ausgangspunkt dieser Arbeit war der Plattformwechsel im Roboterfußball und die Frage, wie der technisch wirkende Booster K1 durch sichtbare Reaktionen mehr Persönlichkeit erhalten kann. Der Torjubel verbindet diese Motivation mit einer konkreten technischen Aufgabe.

Entstanden ist eine Pipeline, die eine Bewegung aus einem Video über B-Human bis auf den realen K1 führt. GVHMR rekonstruiert die menschliche Bewegung, GMR überträgt sie auf den K1 und der K1 Animator macht sie in Blender bearbeitbar. Danach kann die Animation entweder als Keyframe-Geste ausgegeben oder als Referenz für eine Ganzkörper-Policy verwendet werden.

Der K1 Animator bildet den wichtigsten eigenen Werkzeugbeitrag. Er verbindet das K1-Rig, den Import mehrerer Bewegungsformate, die visuelle Bearbeitung und den Export für Isaac Lab und B-Human. Dadurch ist die Pipeline nicht auf den Macarena-Tanz beschränkt. Auch manuell erstellte Animationen können denselben Weg nutzen.

Die EmoteEngine ergänzt die Laufzeitseite. Sie verwaltet Policy- und Keyframe-Emotes in rollenbezogenen Listen für Torschützen und Unterstützer und ordnet sie in den bestehenden Weg von Strategy, Skill und Motion ein. Der Macarena-Tanz zeigt den trainierten Ganzkörperpfad, während der Dab den direkten Weg von einer kurzen Animation zur Keyframe-Ausführung ergänzt. Für weitere einfache Keyframe-Bewegungen dient der K1 Animator als Autorenwerkzeug; die PuppeteerEngine kann die erzeugten Dateien in der Simulation laden und prüfen.

Die Evaluation belegt den funktionalen Ablauf in der Simulation und auf dem realen K1 sowie den code-seitigen Aufbau der Schnittstelle zwischen Training und B-Human. Die Dimensionen, Reihenfolge und Zuordnung der Policy-Ein- und -Ausgaben wurden abgeglichen. Auf dem Roboter zeigten Policy- und Keyframe-Emotes den zuvor in der Simulation beobachteten Bewegungsablauf. Dieser Nachweis bewertet die technische Integration anhand der umgesetzten Beispiele; eine samplegenaue Gleichheit aufgezeichneter Laufzeitwerte oder eine langfristige Erprobung im Spiel wird nicht beansprucht.

8.2 Einordnung der gewählten Lösung

Der Workflow kombiniert automatische und manuelle Schritte. Video-Motion-Capture und Retargeting liefern eine Ausgangsbewegung. Blender bleibt als menschliche Bearbeitungsstufe erhalten, sodass Retargeting-Ergebnisse und gestalterische Änderungen vor dem Training beziehungsweise Export gezielt bearbeitet werden können.

Auch die zwei Ausführungspfade sind eine bewusste Abwägung. Ein RL-Training wäre für jede kleine Geste unnötig aufwendig. Reine Keyframes reichen dagegen nicht für jeden dynamischen Ganzkörperpertanz. Policy- und Keyframe-Pfad erlauben es, den Aufwand an die gewünschte Bewegung anzupassen.

8.3 Nächste Schritte

Ein nächster praktischer Schritt ist die Reduktion des Speicherbedarfs. Insbesondere lange, mit 50 Hz gespeicherte Policy-Referenztrajektorien sind deutlich größer als einzelne Keyframe-Animationen. Zu prüfen sind eine geringere Zahl gespeicherter Referenzframes, eine passende Interpolation zur Laufzeit und eine komprimierte Repräsentation, ohne den sichtbaren Bewegungsablauf oder die zeitliche Zuordnung der Policy zu verändern.

Weiterhin sollte die Trainingsumgebung systematisch an die tatsächlich in B-Human und auf dem K1 verwendeten Motorparameter und Zustandsdaten angenähert werden. Dazu gehören Verstärkung, Dämpfung und Drehmomentgrenzen ebenso wie Sensorfilterung und Verzögerungen. Die in dieser Arbeit eingesetzte Policy funktionierte im Hardwaretest und bildet damit eine praktisch geprüfte Grundlage. Eine genauere Abstimmung kann den Sim-to-Real-Übergang für weitere Bewegungen robuster und nachvollziehbarer machen.

Vor einem dauerhaften Einsatz im aktiven Spielbetrieb bleiben außerdem wiederholte Belastungs-, Abbruch- und Übergangstests unter realistischen Bedingungen erforderlich. Diese Weiterarbeit betrifft die Absicherung und Skalierung der funktionierenden Integration, nicht mehr den grundsätzlichen Nachweis der Pipeline.

Für weitere Tänze kann dieselbe Pipeline wiederverwendet werden. Kurze Gesten lassen sich im K1 Animator als Keyframes erstellen und anschließend in der PuppeteerEngine oder EmoteEngine prüfen. Neue Ganzkörperbewegungen benötigen eine eigene Referenz und ein neues Policy-Training, können danach aber ohne neue Tanzlogik in die EmoteEngine eingetragen werden.

Literaturverzeichnis

- Araujo, J. P., Ze, Y., Xu, P., Wu, J., & Liu, C. K. (2025). Retargeting Matters: General Motion Retargeting for Humanoid Motion Tracking. *arXiv preprint arXiv:2510.02252*. <https://doi.org/10.48550/arXiv.2510.02252>
- B-Human. (2026). *RoboCup 2026*. <https://www.b-human.de/events-en.html>
- Blender Foundation. (2026). *Blender (Version 5.2)*. <https://www.blender.org/>
- BoosterRobotics. (2026a). *booster_assets: Booster Robot Descriptions and Example Motion Data*. https://github.com/BoosterRobotics/booster_assets
- BoosterRobotics. (2026b). *booster_train: Reinforcement Learning Tasks for Booster Robots using Isaac Lab*. https://github.com/BoosterRobotics/booster_train
- Liao, Q., Truong, T. E., Huang, X., Gao, Y., Tevet, G., Sreenath, K., & Liu, C. K. (2025). BeyondMimic: From Motion Tracking to Versatile Humanoid Control via Guided Diffusion. *arXiv preprint arXiv:2508.08241*. <https://doi.org/10.48550/arXiv.2508.08241>
- Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., & Black, M. J. (2015). SMPL: A Skinned Multi-Person Linear Model. *ACM Transactions on Graphics (TOG)*, 34(6), 248:1–248:16. <https://doi.org/10.1145/2816795.2818013>
- Mittal, M., Roth, P., Tigue, J., Richard, A., Zhang, O., Du, P., Serrano-Muñoz, A., Yao, X., Zurbrugg, R., Rudin, N., Wawrzyniak, L., Rakhsha, M., Denzler, A., Heiden, E., Borovicka, A., Ahmed, O., Akinola, I., Anwar, A., Carlson, M. T., ... State, G. (2025). Isaac Lab: A GPU-Accelerated Simulation Framework for Multi-Modal Robot Learning. *arXiv preprint arXiv:2511.04831*. <https://doi.org/10.48550/arXiv.2511.04831>
- Peng, X. B., Abbeel, P., Levine, S., & van de Panne, M. (2018). DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *ACM Transactions on Graphics (TOG)*, 37(4), 143:1–143:14. <https://doi.org/10.1145/3197517.3201311>
- Peng, X. B., Andrychowicz, M., Zaremba, W., & Abbeel, P. (2018). Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 3803–3810. <https://doi.org/10.1109/ICRA.2018.8460528>
- RoboCup Federation. (2026). *Humanoid Soccer League: Call for Participation*. <https://robocup.org/news/187>
- Röfer, T., Laue, T., Böse, F., Dehmlow, K., Gittner, L., Hasselbring, A., Lienhoop, J., Lührsén, A., Meinken, Y., Monnerjahn, L. M., Plecher, L., Reichenberg, P., Schiefelbein, L., & Schreiber, S. (2024). *B-Human Code Release Documentation 2024*. <https://docs.b-human.de/coderelease2024/>
- Röfer, T., Laue, T., Reichenberg, P., Gittner, L., & Schiefelbein, L. (2025). B-Human 2024 – Enhanced Vision and Faster Ball Handling. *RoboCup 2024: Robot World Cup XXVII, Lecture Notes in Computer Science*, 15570, 483–494. https://doi.org/10.1007/978-3-031-85859-8_41
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*. <https://doi.org/10.48550/arXiv.1707.06347>
- Shen, Z., Pi, H., Xia, Y., Cen, Z., Peng, S., Hu, Z., Bao, H., Hu, R., & Zhou, X. (2024). World-Grounded Human Motion Recovery via Gravity-View Coordinates. *SIGGRAPH Asia Conference Proceedings*, 144:1–144:11. <https://doi.org/10.1145/3680528.3687565>
- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2. Aufl.). The MIT Press. <https://mitpress.mit.edu/9780262039246/reinforcement-learning/>

A Nutzung KI-basierter Anwendungen

Bei der Entwicklung der Software und beim Schreiben dieser Arbeit wurden generative KI-Werkzeuge als Assistenz eingesetzt. Sie halfen bei der Analyse und Überarbeitung von Code, bei der Fehlersuche, bei der Recherche sowie bei der sprachlichen Verbesserung bereits geschriebener Texte.

A.1 Verwendete Werkzeuge

Anbieter oder Dienst	Verwendete Modelle	Schwerpunkt der Nutzung
OpenAI	GPT-5.6-sol	sprachliche Überarbeitung, Verständlichkeitsprüfung und Fehlersuche im Text
Anthropic	Claude Sonnet 5 und Fable 5	Programmierung, Codeanalyse, Fehlersuche und Optimierung
Google	Gemini 3.1 Pro und Gemini 3.5 Flash	Programmierung, Codeanalyse und Fehlersuche
Perplexity	Perplexity	Recherche und Auffinden möglicher Quellen

A.2 Art der Unterstützung

Bei der Softwareentwicklung dienten die Werkzeuge dazu, vorhandene Quelltexte und Schnittstellen schneller zu verstehen, Fehlermeldungen einzuordnen, Implementierungen zu überarbeiten und Inkonsistenzen zwischen Blender, Isaac Lab und B-Human zu finden. Übernommene Vorschläge wurden an das Gesamtsystem angepasst und anhand des Quellcodes und der beteiligten Systeme geprüft.

Beim Schreiben halfen die Werkzeuge, unklare Formulierungen, Widersprüche, Wiederholungen und fehlende Erklärungen zu erkennen. Perplexity wurde ergänzend zur Recherche eingesetzt. KI-Ausgaben wurden nicht als wissenschaftliche Quellen verwendet; relevante Angaben wurden anhand der jeweiligen Originalquellen geprüft.

A.3 Eigenanteil und Verantwortung

Fragestellung, Zielsetzung, Grundidee und Aufbau der Pipeline wurden vom Verfasser entwickelt. Dies gilt ebenso für die fachliche Logik, die zentralen Architekturentscheidungen, den K1 Animator, die B-Human-Integration, die Auswahl und Durchführung der Beispiele sowie Aufbau und inhaltliche Argumentation der Arbeit. Das Projekt umfasst mehrere hundert Stunden eigener Arbeit an Code, Animationen, Training, Integration und Dokumentation.

Die KI-Werkzeuge hatten eine unterstützende Funktion. Einzelne Code- und Formulierungsvorschläge wurden geprüft, angepasst oder verworfen. Die fachlichen Entscheidungen sowie die Verantwortung für den finalen Code und den Inhalt der Arbeit blieben beim Verfasser.

B Struktur einer Keyframe-Animation

Die Keyframe-Emotes verwenden eine gemeinsame Animationsstruktur. Der K1 Animator kann diese Dateien erzeugen; EmoteEngine und PuppeteerEngine können sie laden. Eine Datei besitzt einen Namen und eine Folge von Frames. Jeder Frame enthält eine Dauer, einen Steifigkeitswert und die Gelenkwinkel des K1.

Ebene	Feld	Bedeutung
Animation	name	eindeutiger Name der Bewegung
Animation	frames	zeitlich geordnete Folge der Posen
Frame	duration	Interpolationsdauer in Millisekunden von diesem Frame zum folgenden
Frame	angles	Gelenkwinkel der Pose
Frame	stiffness	Steifigkeitswert für die Ausführung

Tabelle 12: Struktur einer Keyframe-Animation für die EmoteEngine

Die aktuelle Konfiguration enthält sechs solche Keyframe-Emotes:

Emote	Frames	Summe duration
clap	231	7.623 ms
dab	93	3.069 ms
wave	213	7.029 ms
sweep	176	5.808 ms
groove	288	5.760 ms
sixseven	163	3.260 ms

Tabelle 13: Keyframe-Emotes im aktuellen B-Human-Konfigurationsstand

Bei der Ausführung übernimmt die EmoteEngine Kopf und Arme aus den Frames, während der StandGenerator die Beine führt. K1 Animator, EmoteEngine und PuppeteerEngine verwenden dieselbe Zeitsemantik: **duration** beschreibt jeweils das Intervall vom aktuellen Frame zum folgenden. Nach dem separaten Einstieg interpoliert die EmoteEngine entsprechend zum nächsten Frame und am Ende auch in Richtung des ersten Frames. Die PuppeteerEngine führt dieselben Frame-Übergänge aus und nutzt die Steifigkeit des aktuellen Frames. Bei einer Schleife beginnt anschließend erneut das Intervall des ersten Frames; ohne Schleife endet die Wiedergabe nach genau einem vollständigen Durchlauf. Die EmoteEngine bleibt für die Laufzeitausführung maßgeblich, während die PuppeteerEngine als Prüfwerkzeug in SimRobot dient.

C Schnittstellenübersicht der Beispiele

Die folgende Übersicht fasst die Dateien der beiden durchlaufenden Beispiele zusammen. Macarena nutzt nach Blender den Policy-Pfad, während der Dab direkt als Keyframe-Animation nach B-Human gelangt.

Datei	Erzeuger	Verbraucher und Zweck
MP4-Video	Aufnahme	GVHMR als Bewegungsvorlage für Macarena
PT-Datei	GVHMR	GMR mit rekonstruierter menschlicher Bewegung
PKL-Datei	GMR	K1 Animator für Import und Bearbeitung
Blender-Action	K1 Animator	gemeinsame visuelle Bewegungsdarstellung
NPZ	K1 Animator	Isaac Lab als Tracking-Referenz für Macarena
ONNX	Isaac Lab	EmoteEngine als Macarena-Policy
Trajektorien-CFG	K1 Animator	EmoteEngine als Macarena-Referenz
Animations-CFG	K1 Animator	EmoteEngine als Dab-Keyframe

Tabelle 14: Dateien und Schnittstellen der durchlaufenden Beispiele

Die Laufzeitgrenze liegt zwischen den letzten drei Dateien. B-Human benötigt für die Macarena-Policy das ONNX-Modell und die Trajektorien-CFG; die vollständige NPZ-Datei wird dort nicht geladen. Der Dab verwendet nur die Animations-CFG und kein neuronales Netz.

Die feste Zuordnung verhindert, dass gleich benannte Formate als austauschbar missverstanden werden. Insbesondere enthalten die beiden CFG-Varianten unterschiedliche fachliche Daten: Die Policy-Trajektorie liefert Referenzpositionen und -geschwindigkeiten in 20-ms-Schritten, während die Keyframe-Datei Posen mit individuellen Dauern und Steifigkeiten beschreibt.