POLITECNICO DI TORINO

Master's Degree in Ingegneria del cinema e dei mezzi di comunicazione

Master's Degree Thesis

Generazione di posture umane tramite deep reinforcement learning per la simulazione ergonomica di postazioni di lavoro industriali.

Supervisori

Prof. Andrea SANNA

Prof. Federico MANURI

Tutor. Sebastiano LAMACCHIA

Candidato

Gian Marco LELLI

Aprile 2025

Sommario

La progettazione delle postazioni di lavoro in Italia deve seguire rigorosamente le normative stabilite dal D.Lgs. 81/2008 [1] e successive integrazioni, con particolare attenzione all'ergonomia, alla sicurezza e al comfort ambientale. Questi requisiti non solo sono fondamentali per tutelare la salute dei lavoratori, ma permettono anche di aumentare l'efficienza operativa nelle industrie. Pertanto, migliorare il design ergonomico delle postazioni di lavoro rappresenta una priorità in ambito industriale.

Numerose figure professionali sono coinvolte nel processo. Designer ed ergonomi collaborano per creare postazioni di lavoro funzionali e sicure. Tuttavia, la comunicazione tra questi esperti non è sempre chiara ed efficace, causando disallineamenti in fase di progettazione. Questo si traduce nella realizzazione di prototipi fisici che necessitano di revisioni e modifiche, con un conseguente aumento dei costi di sviluppo. Inoltre, postazioni di lavoro progettate in modo non ergonomico possono causare insoddisfazione tra gli utilizzatori e problemi muscoloscheletrici, con ripercussioni sia sulla produttività aziendale sia sui costi legati alla salute dei dipendenti.

Per prevenire la messa in produzione di postazioni di lavoro non ottimali, si ricorre a tecnologie avanzate che consentono una valutazione più precisa dal punto di vista progettuale ed ergonomico. L'uso della VR (realtà virtuale) permette di simulare e analizzare l'efficacia delle postazioni prima della loro realizzazione, offrendo un riscontro immediato sulle soluzioni adottate. Inoltre, grazie ai modelli basati su AI (intelligenza artificiale), è possibile replicare i movimenti umani, consentendo di valutare l'impatto ergonomico e ottimizzare le condizioni di lavoro. Questi strumenti non solo riducono il rischio di errori in fase di progettazione, ma facilitano anche una collaborazione più efficace tra i diversi professionisti, grazie all'adozione di un linguaggio visivo condiviso.

Alla luce di quanto appena esposto, il presente lavoro si propone di sviluppare un software che utilizzi modelli di AI, per generare movimenti umani realistici all'interno di workstation virtuali. Con l'uso di queste tecnologie, si vuole minimizzare la necessità della prototipazione fisica delle postazioni di lavoro, abbattendo i costi di produzione e accelerando i tempi di sviluppo.

Indice

Elenco delle figure

Acronimi

VR

La realtà virtuale (Virtual Reality) è una tecnologia che simula ambienti tridimensionali attraverso dispositivi informatici, creando esperienze immersive per l'utente.

AI

L'intelligenza artificiale (Artificial Intelligence) è definita come un campo dell'informatica incentrato sulla creazione di sistemi in grado di eseguire compiti che richiedono tipicamente l'intelligenza umana.

ML

L'apprendimento automatico (Machine Learning) è un sottoinsieme dell'intelligenza artificiale che si concentra sulla creazione di algoritmi e modelli statistici che consentono ai computer di apprendere dai dati.

ANN

Una rete neurale artificiale (Artificial Neural Network) è un modello computazionale ispirato al funzionamento del cervello umano, composto da unità chiamate neuroni artificiali. Queste reti sono utilizzate per risolvere problemi complessi attraverso l'apprendimento automatico, emulando il modo in cui i neuroni biologici interagiscono tra loro.

DNN

Le reti neurali profonde (Deep Neural Networks) sono un tipo di rete neurale artificiale caratterizzata dalla presenza di più strati nascosti, che consentono di apprendere rappresentazioni complesse e astratte dei dati. Queste reti sono una sottocategoria del deep learning, una branca dell'apprendimento automatico che emula il funzionamento del cervello umano attraverso nodi interconnessi, chiamati neuroni.

RL

L'apprendimento per rinforzo (Reinforcement Learning)è una tecnica di apprendimento automatico che consente a un agente di apprendere come compiere azioni in un ambiente per massimizzare una ricompensa cumulativa. Questo approccio è caratterizzato dall'interazione continua tra l'agente e l'ambiente, dove l'agente esplora diverse azioni e riceve feedback sotto forma di ricompense o penalità a seconda delle sue scelte.

DRL

L'apprendimento per rinforzo profondo (Deep Reinforcement Learning) è un approccio nell'ambito dell'intelligenza artificiale che combina tecniche di apprendimento per rinforzo con reti neurali profonde. Questo metodo consente a un agente di apprendere a prendere decisioni ottimali in ambienti complessi e dinamici attraverso l'interazione diretta con l'ambiente stesso.

AR

La realtà aumentata (Augmented Reality) è una tecnologia che sovrappone elementi digitali all'ambiente reale, creando un'esperienza interattiva che arricchisce la percezione sensoriale dell'utente. Essa integra informazioni digitali, come testi, immagini e modelli 3D, nel mondo fisico, permettendo agli utenti di interagire con entrambi gli ambienti in modo simultaneo.

SMPL

Il modello Skinned Multi-Person Linear Model è una rappresentazione 3D realistica del corpo umano, sviluppata per essere utilizzata in animazione e computer grafica.

FBX

Il formato Filmbox è una tipologia di file proprietario sviluppato originariamente da Kaydara e attualmente di proprietà di Autodesk. È ampiamente utilizzato nell'industria della grafica 3D per facilitare l'interoperabilità tra diverse applicazioni di creazione di contenuti digitali, consentendo la memorizzazione e lo scambio di modelli 3D, animazioni e altri asset digitali.

JSON

JavaScript Object Notation è un formato di interscambio dati basato su testo, utilizzato per rappresentare strutture di dati in modo semplice e leggibile. È progettato per essere facilmente comprensibile sia dagli esseri umani che dalle macchine, rendendolo ideale per la trasmissione di dati tra client e server in applicazioni web.

BVH

BioVision Hierarchy è un formato di file utilizzato per la rappresentazione di dati di animazione 3D, in particolare per descrivere la gerarchia e il movimento di scheletri virtuali. Questo formato è comunemente impiegato nel campo della grafica computerizzata e dell'animazione per facilitare l'importazione e l'esportazione di dati di motion capture.

LBS

Linear Blend Skinning è un algoritmo di deformazione utilizzato per animare modelli 3D, in particolare nel contesto della grafica computerizzata e dei videogiochi. Questo metodo consente di attaccare una mesh (la pelle) a uno scheletro virtuale, permettendo alla mesh di deformarsi in modo fluido quando lo scheletro si muove.

Capitolo 1

Introduzione

Dalla prima rivoluzione industriale, avvenuta alla fine del XVIII secolo, fino alla quarta, l'ambiente produttivo ha subito trasformazioni significative, caratterizzate dall'evoluzione delle tecnologie di fabbricazione. Nonostante i progressi tecnologici, l'uomo continua a ricoprire un ruolo centrale nei sistemi produttivi, sebbene le sue mansioni siano cambiate nel tempo. Le tecnologie impiegate **nell'Industria 4.0**, fulcro della quarta rivoluzione industriale, hanno reso i processi produttivi più efficienti, ma al contempo hanno introdotto nuove sfide per i lavoratori, influenzando il loro benessere e il loro modo di operare. Di conseguenza, lo **studio ergonomico** applicato ai metodi di lavoro e alla progettazione delle postazioni è oggi più rilevante che mai. L'obiettivo è migliorare sia la salute e il comfort degli operatori sia le prestazioni del sistema produttivo, analizzando l'interazione tra l'uomo e gli altri elementi dell'impianto. Per raggiungere questo scopo, si utilizzano **simulazioni virtuali** che riproducono i processi produttivi, permettendo di valutare il comportamento degli operatori e di progettare in modo ottimale le postazioni di lavoro. Il supporto di software di simulazione 3D consente di individuare criticità e ottimizzare le soluzioni prima della loro implementazione fisica. In questo capitolo verranno approfonditi i concetti di **Industria 4.0** ed **ergonomia**, evidenziando come l'uso di sistemi virtuali possa supportare la progettazione ergonomica delle workstation. Inoltre, verrà fornita una panoramica sul **ML (machine learning)** e sugli algoritmi esistenti, utili per la comprensione dei capitoli successivi.

1.1 Industria 4.0

Fin dall'inizio della rivoluzione industriale, i professionisti hanno costantemente ricercato soluzioni innovative per ottimizzare i processi produttivi, migliorandone efficienza, riduzione dei costi e qualità del prodotto. Per soddisfare questi requisiti, l'industria ha attraversato quattro fasi evolutive principali come illustrato in Figura 1.1. La prima rivoluzione industriale (fine XVIII - inizio XIX secolo) ha introdotto la meccanizzazione a vapore e ad acqua, segnando il passaggio dalla produzione artigianale a quella industriale. Successivamente, la seconda rivoluzione industriale (fine XIX - inizio XX secolo) ha rivoluzionato il settore con l'elettrificazione e la produzione di massa, trasformando radicalmente i metodi di fabbricazione su larga scala. Negli anni Settanta del XX secolo ha avuto inizio la terza rivoluzione industriale, caratterizzata dall'introduzione della robotica e delle tecnologie digitali, con l'obiettivo di migliorare l'efficienza e la flessibilità dei processi produttivi. Infine, con la quarta rivoluzione industriale, avviata nel 2011, si è vista l'integrazione di sistemi ciber-fisici e dell'AI (intelligenza artificiale), che hanno dato origine a fabbriche intelligenti e a processi completamente automatizzati e interconnessi [2].

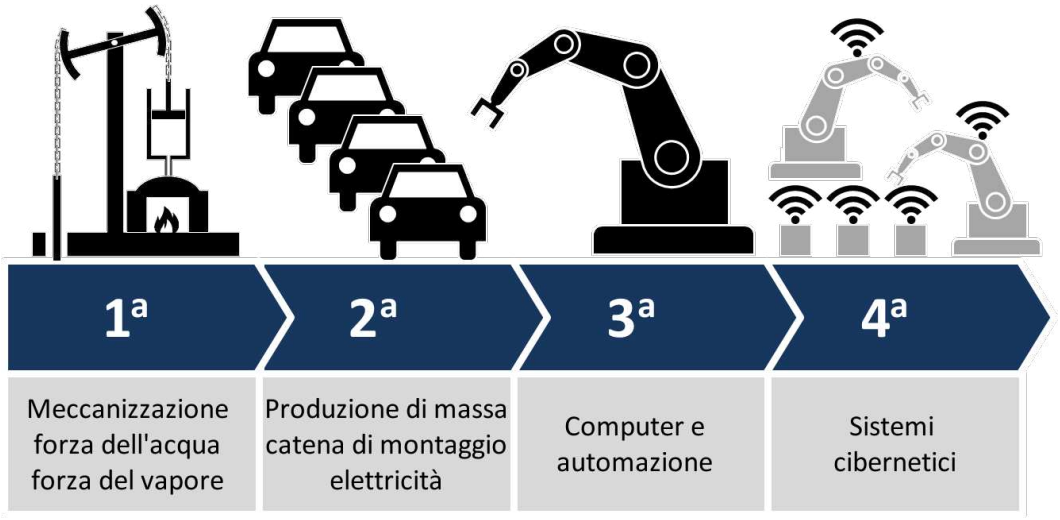


Figura 1.1: Cronologia delle Rivoluzioni Industriali, dalla prima che ebbe inizio alla fine del 1700 fino ai giorni odierni.

Questa evoluzione ha suscitato un forte interesse sia in ambito accademico che industriale, a tal punto che nel 2011, un gruppo di esperti provenienti da diversi settori in Germania ha introdotto il concetto di Industria 4.0 [3].

Il termine **Industria 4.0** fa riferimento a un nuovo modello produttivo basato sull'integrazione di **tecnologie avanzate** all'interno dei processi industriali, con l'obiettivo di trasformare le fabbriche in ambienti **altamente automatizzati**,

interconnessi e **intelligenti**. Questo paradigma apre a numerose opportunità di ricerca, poiché introduce sistemi complessi e soluzioni tecnologiche eterogenee all'interno di contesti operativi in cui l'essere umano continua a svolgere un ruolo attivo. Tra le aree di indagine più rilevanti si colloca quella dell'interazione uomo-macchina, che mira a garantire ambienti di lavoro sicuri, ergonomici ed efficienti.

All'interno di questa tesi, Industria 4.0 rappresenta il contesto ideale per lo sviluppo di un sistema di simulazione virtuale finalizzato allo studio e al miglioramento dell'ergonomia delle postazioni di lavoro. In tale prospettiva, risulta essenziale approfondire le tecnologie abilitanti che definiscono questo modello industriale, nonché le applicazioni attualmente in uso e le prospettive future legate alla loro integrazione nei processi produttivi. I paragrafi seguenti offriranno una panoramica sullo stato dell'arte di tali tecnologie, esaminando le tendenze emergenti e le direzioni di sviluppo più promettenti. A seguire, l'attenzione verrà posta sul concetto di ergonomia, analizzato come elemento chiave per la progettazione di ambienti di lavoro che siano al contempo sicuri, funzionali ed efficienti.

1.1.1 Tecnologie abilitanti

Le tecnologie che caratterizzano l'Industria 4.0 comprendono:

- **Sistemi ciberfisici (CPS)**: soluzioni fondamentali per l'automazione intelligente in quanto permettono di effettuare **manutenzione predittiva e il controllo in tempo reale** dei processi produttivi.
- **Internet of things (IoT)**: rete di dispositivi connessi che consente la **raccolta dati in tempo reale**, facilitando la comunicazione tra le diverse componenti del sistema produttivo attraverso Internet.
- **Cloud computing**: modello di erogazione di risorse informatiche che offre **capacità di elaborazione, archiviazione e analisi dei dati on-demand**.
- **Big data**: enormi quantità di dati raccolti da diverse fonti industriali. L'analisi avanzata di **Big Data**, tramite tecniche di data mining e machine learning, permette di **estrarre informazioni utili** per il processo decisionale, l'ottimizzazione produttiva e lo sviluppo di nuove soluzioni industriali.
- **AI (intelligenza artificiale)**: algoritmi avanzati che consentono ai sistemi di **apprendere dai dati e migliorare le loro prestazioni nel tempo**.

È fondamentale sottolineare che queste tecnologie non operano in modo indipendente, ma si integrano tra loro, per creare un **ecosistema connesso e dinamico**, in grado di evolversi continuamente.

1.1.2 Applicazioni dell'Industria 4.0 nei diversi settori industriali

Le tecnologie abilitanti dell'Industria 4.0 hanno trovato ampia applicazione in vari settori, trasformando i processi produttivi e migliorando l'efficienza operativa. Alcuni esempi significativi includono:

- **Gemelli digitali (Digital Twins)**: **repliche virtuali** di ambienti produttivi fisici [4]. Un esempio concreto è **Siemens**, che utilizza questa tecnologia per simulare processi di produzione, eseguire analisi predittive e ottimizzare l'utilizzo delle risorse, contribuendo alla riduzione dei costi operativi.
- **Robot collaborativi (Cobot)**: che lavorano in stretta sinergia con gli operatori umani. **BMW**, ad esempio, ha implementato cobot nelle sue linee di assemblaggio, sfruttando algoritmi di intelligenza artificiale per apprendere movimenti ottimali e adattarsi alle esigenze degli operatori [5].
- **Big Data e manutenzione predittiva**: ampiamente utilizzata nel settore automobilistico da aziende come **Tesla**, che sfrutta l'analisi dei dati in tempo reale raccolti dai sensori IoT per individuare anomalie nei sistemi e prevenire guasti [6].
- **Progetti accademici e ricerca**: un esempio rilevante è il **framework "Smart Factory KL"** sviluppato in Germania [7]. Questo progetto dimostra come l'integrazione di CPS, IoT e AI possa generare fabbriche intelligenti capaci di adattarsi dinamicamente ai cambiamenti della domanda e delle configurazioni produttive. Inoltre, il framework incorpora simulazioni ergonomiche, contribuendo a migliorare la sicurezza e il comfort degli operatori.

Questi esempi evidenziano come l'Industria 4.0 non si limiti a migliorare l'efficienza produttiva, ma abbia un impatto significativo anche sull'interazione uomo-macchina. L'integrazione di tecnologie avanzate non solo ottimizza i processi industriali, ma contribuisce anche a creare ambienti di lavoro più sicuri, adattivi ed ergonomici, favorendo il benessere degli operatori e aumentando la produttività complessiva.

1.2 Introduzione all'ergonomia

L'**ergonomia** è una disciplina scientifica che studia l'interazione tra gli esseri umani e gli elementi di un sistema, con l'obiettivo di **ottimizzare il benessere delle persone e migliorare le prestazioni complessive**. Il termine deriva dal greco *ergon* (lavoro) e *nomos* (legge), evidenziando il suo ruolo nella progettazione di ambienti, strumenti e processi lavorativi che si adattino alle esigenze umane. Grazie alla sua natura interdisciplinare, integra conoscenze provenienti da campi come psicologia, anatomia, fisiologia, fisica, ingegneria, neuroscienze e intelligenza artificiale [8].

Il ruolo ricoperto da questa materia scientifica è fondamentale nel migliorare la sicurezza e la produttività nei luoghi di lavoro, contribuendo a ridurre gli infortuni, ottimizzare l'efficienza operativa e garantire il benessere dei lavoratori. L'applicazione di metodologie ergonomiche sul posto di lavoro ha dimostrato un impatto significativo sulla riduzione dei disturbi muscolo-scheletrici (DMS) [8]. Studi evidenziano che l'implementazione di programmi ergonomici mirati può portare a una riduzione media del 59% dei DMS [9].

Oltre agli aspetti legati alla salute, l'adozione di soluzioni ergonomiche efficaci ha un impatto diretto sulla produttività. Secondo alcune ricerche, una progettazione ergonomica adeguata può portare a un aumento della produttività fino al 25% [10]. Inoltre, migliorare le condizioni ergonomiche influisce positivamente anche sulla riduzione dell'assenteismo e del turnover del personale, poiché i lavoratori in buona salute tendono a rimanere più a lungo in azienda e a richiedere meno giorni di malattia [9].

Alla base della filosofia progettuale dietro un sistema ergonomicamente corretto è centrale il rispetto dei bisogni e degli interessi dell'utente, con l'obiettivo di creare prodotti e sistemi facili da usare e intuitivi [11]. Questo paradigma è definito con il termine **design centrato sull'utente**, e la sua applicazione nei contesti produttivi mira a **migliorare la soddisfazione dell'utente**, aumentare l'efficienza d'uso, garantire il comfort e ridurre i rischi legati all'utilizzo improprio di strumenti e ambienti di lavoro [11, 12]. I principi su cui si basa il design centrato sull'essere umano sono:

- **Centralità sulle persone**: Concentrarsi sulle persone e sul loro contesto per creare soluzioni adatte a loro.
- **Comprensione e risoluzione di problemi giusti, cercando le cause alla radice**: È fondamentale individuare e affrontare le cause profonde dei problemi, altrimenti i sintomi continueranno a ripresentarsi.
- **Complessività del sistema**: Ogni elemento deve essere considerato come parte di un sistema interconnesso.

- **Gestione dei problemi attraverso interventi piccoli e semplici**: Lavorare in modo iterativo senza affrettarsi verso una soluzione. Provare piccoli interventi semplici e imparare da essi uno alla volta. Nel tempo, i risultati diventeranno sempre più grandi e significativi. Prototipare, testare e perfezionare continuamente le proposte per garantire che le soluzioni rispondano veramente alle esigenze delle persone.

Focalizzarsi sull'aspetto umano della progettazione ergonomica consente non solo di migliorare l'esperienza degli utenti, ma anche di sviluppare sistemi più sostenibili e adattabili. Con l'aumento della complessità dei sistemi socio-tecnici e delle sfide globali, il design centrato sull'essere umano diventa sempre più essenziale per affrontare le trasformazioni dell'industria e migliorare l'interazione tra tecnologia e persone [13].

1.2.1 L'evoluzione dell'ergonomia nell'industria

L'evoluzione dell'**ergonomia nell'industria** è caratterizzata da un progressivo passaggio da un approccio incentrato sull'individuo a una visione più ampia che considera **l'organizzazione e i sistemi nel loro complesso**. Questo cambiamento riflette la crescente complessità dei contesti lavorativi e l'integrazione di nuove tecnologie nei processi produttivi [14].

Questa disciplina nasce per risolvere problemi complessi e multidisciplinari derivanti dai progressi tecnologici del XX secolo. Figure come Frederick Taylor e Frank e Lillian Gilbreth, con i loro studi sulla gestione scientifica del lavoro e sull'ottimizzazione dei movimenti, hanno gettato le basi della disciplina, inizialmente focalizzata sull'aumento della produttività attraverso la riprogettazione delle mansioni lavorative [8]. Successivamente, il campo di applicazione si è ampliato, includendo non solo l'adattamento di attrezzature e spazi di lavoro alle caratteristiche umane, ma anche aspetti psicologici, sociali e organizzativi del lavoro [8, 11]. Questo ha portato allo sviluppo di tre principali aree di specializzazione: **ergonomia fisica, cognitiva e organizzativa** [15, 16].

Con il progresso tecnologico, l'ergonomia ha integrato strumenti innovativi per migliorare la progettazione degli ambienti di lavoro. L'introduzione di tecnologie come la VR (realtà virtuale) ha aperto nuove possibilità nella simulazione e valutazione delle postazioni di lavoro, consentendo la creazione di prototipi digitali e l'analisi dell'impatto di diverse configurazioni sull'efficienza, sulla sicurezza e sul benessere dei lavoratori [17]. In parallelo, dispositivi indossabili e sensori avanzati permettono di raccogliere dati biometrici e posturali in tempo reale, migliorando l'affidabilità delle valutazioni ergonomiche [15].

In aggiunta, con l'avvento dell'Industria 4.0 questa evoluzione è ulteriormente accellerata, introducendo un nuovo modello produttivo basato su automazione, digitalizzazione e interconnessione. In questo contesto, l'ergonomia assume un ruolo

cruciale, spostando il focus verso un **approccio macroergonomico** che considera non solo il singolo lavoratore, ma l'intero sistema socio-tecnico. Questo comporta un'analisi approfondita dell'interazione tra uomo e tecnologia, dell'organizzazione del lavoro e dei fattori psicosociali [14].

Nel corso del tempo, sono stati sviluppati numerosi strumenti e metodi per l'analisi e la valutazione dei rischi ergonomici. Tra i più utilizzati vi sono checklist ergonomiche [8], analisi delle attività lavorative e sistemi di valutazione rapida come RULA (Rapid Upper Limb Assessment) e REBA (Rapid Entire Body Assessment) [18]. L'uso di questi strumenti consente di identificare rapidamente le criticità ergonomiche e di proporre interventi correttivi mirati.

Un aspetto importante che si è introdotto durante l'evoluzione dell'ergonomia è il crescente coinvolgimento dei lavoratori nel processo di progettazione ergonomica. L'approccio partecipativo si è rivelato essenziale per garantire l'efficacia e l'accettazione delle soluzioni proposte, migliorando il comfort e la sicurezza percepita dagli operatori. Grazie a questo metodo si ha un cambio di paradigma che non si limita più a un intervento top-down, ma si basa su un dialogo continuo tra progettisti, manager e lavoratori, in un'ottica di miglioramento collaborativo [8, 19].

Parallelamente, alla continua evoluzione dell'ergonomia, si è assistito a una progressiva integrazione all'interno di normative e standard internazionali. Organizzazioni come l'ISO (International Organization for Standardization) e il CEN (Comitato Europeo di Normazione) hanno sviluppato linee guida per la progettazione ergonomica di prodotti, ambienti e sistemi di lavoro [11, 15]. A livello nazionale, molte normative riflettono l'importanza crescente dell'ergonomia nella tutela della salute dei lavoratori: in Italia, il D.Lgs. 626/94 ha riconosciuto ufficialmente il ruolo dell'ergonomia nella prevenzione dei rischi professionali, anticipando disposizioni successive in materia di sicurezza sul lavoro [11].

Nel tempo l'ergonomia continua ad evolversi per affrontare le sfide poste dalle nuove tecnologie e dai cambiamenti nel mondo del lavoro. Tuttavia, l'obiettivo rimane sempre quello di creare sistemi di lavoro sicuri, efficienti e sostenibili, garantendo un equilibrio tra innovazione tecnologica e benessere umano [8].

1.2.2 Tecnologie per l'ergonomia industriale

Nell'ergonomia industriale, l'impiego di tecnologie avanzate gioca un ruolo fondamentale nel migliorare la sicurezza, l'efficienza e il benessere dei lavoratori. L'adozione di strumenti innovativi consente di monitorare e ottimizzare le condizioni di lavoro, riducendo il rischio di infortuni e migliorando l'interazione tra operatore e ambiente produttivo. Tra le tecnologie più rilevanti troviamo:

- **Dispositivi indossabili**: sistemi di sensori posturali, applicati in diverse parti del corpo, permettono di identificare posture scorrette e valutare il rischio

biomeccanico associato ai movimenti ripetitivi. Strumenti come smartwatches e smartphone montati sul corpo forniscono dati dettagliati sull'esposizione a carichi fisici e posture scorrette. Inoltre, interfacce di feedback vibrotattile offrono un segnale immediato quando viene rilevata una posizione rischiosa, consentendo al lavoratore di correggere la postura in tempo reale [15].

- **VR (realtà virtuale)**: tecnologia per la progettazione e il testing di ambienti di lavoro in modo virtuale, riducendo la necessità di costosi mock-up fisici. Questa categoria include la VR immersiva e le tecnologie aptiche che permettono di simulare in modo realistico le interazioni tra operatore e postazione di lavoro, valutando in anticipo eventuali criticità ergonomiche. Questo approccio facilita la prototipazione virtuale di postazioni produttive, migliorando l'ergonomia delle attività operative e ottimizzando l'efficienza dei processi [19, 17].

- **AI (intelligenza artificiale)** e le tecniche di **ML (machine learning)**: algoritmi per l'automatizzazione e il miglioramento dei metodi di analisi posturale, che permettono di ridurre il margine di errore associato alle valutazioni manuali [18].

- **Robotica collaborativa**: sistemi che introducono robot intelligenti in grado di interagire con gli operatori. Questi, sono progettati per ridurre il carico biomeccanico delle attività ripetitive o faticose, migliorando il benessere fisico dei lavoratori [16].

È importante sottolineare che l'efficacia di queste tecnologie dipende dalla loro corretta integrazione nel sistema di lavoro [15, 14]. Per ottenere risultati concreti, è necessario considerare non solo gli aspetti tecnici, ma anche i fattori umani, organizzativi e ambientali, garantendo una progettazione ergonomica che sia realmente incentrata sulle esigenze dei lavoratori e dell'ambiente produttivo.

1.2.3 Ergonomia Virtuale e Modelli di Simulazione

La **VR (realtà virtuale)**, **AR (realtà aumentata)** con software di progettazione 3D in combinazione con dispositivi tattili, hanno cambiato il modo in cui vengono condotti i test ergonomici. Questi strumenti offrono vantaggi significativi nella valutazione ergonomica, migliorando la produttività, la flessibilità e riducendo i costi, rendendo i processi di progettazione più rapidi ed efficienti [17]. Un esempio di come le nuove tecnologie influenzano il processo di progettazione delle postazioni di lavoro, è illustrato in Figura 1.2.

L'impiego della VR consente ai progettisti di **esplorare soluzioni ergonomiche all'interno di ambienti simulati**, individuando e correggendo potenziali

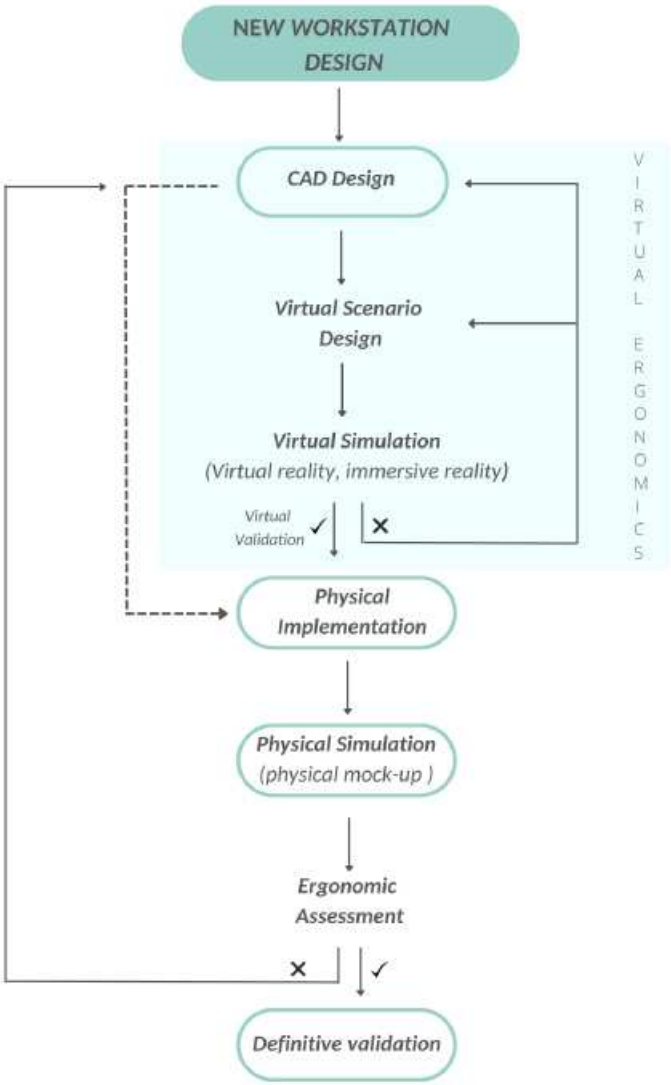


Figura 1.2: Flusso di progettazione di una nuova postazione di lavoro, che integra ergonomia virtuale e validazione fisica. Il processo inizia con la progettazione CAD, seguita dalla simulazione in realtà virtuale per una prima validazione ergonomica. Se approvata, si passa all'implementazione fisica e alla simulazione con mock-up, culminando nella valutazione ergonomica e nella validazione definitiva. [20].

problematiche nelle fasi iniziali dello sviluppo. Grazie a questi strumenti, è possibile testare diverse configurazioni di postazioni di lavoro, attrezzature e prodotti in un ambiente controllato, aumentando la flessibilità e riducendo la necessità

di realizzare prototipi fisici durante la progettazione. Inoltre, permette di facilitare l'**individuazione precoce dei problemi ergonomici**, sia nella fase di progettazione sia nella prevenzione di scenari potenzialmente pericolosi.

Una delle applicazioni più rilevanti della VR riguarda la progettazione di postazioni di lavoro ergonomiche basate sulle caratteristiche antropometriche degli utenti, aiutando a prevenire rischi come posture scorrette e sforzi ripetitivi, come mostrato in Figura 1.3.

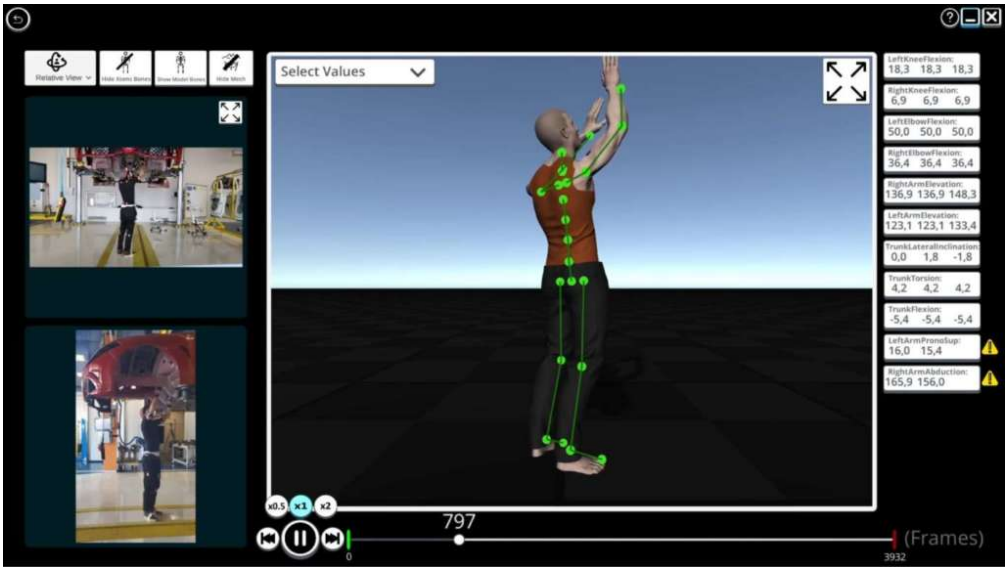


Figura 1.3: Analisi ergonomica di un'operazione di sollevamento in ambiente industriale tramite simulazione digitale. L'immagine mostra un confronto tra un lavoratore reale e un modello biomeccanico virtuale, con tracciamento dei movimenti e misurazione degli angoli articolari [21].

Inoltre, la **valutazione del rischio** è notevolmente agevolata, poiché è possibile ricostruire e simulare scenari pericolosi e sviluppare strategie di mitigazione [15]. Oltre alla progettazione, la VR trova spazio anche in ambito **educativo**, coinvolgendo attivamente i lavoratori nel processo di riprogettazione delle loro postazioni [19].

Grazie ad un approccio interdisciplinare e all'integrazione di tecnologie avanzate, l'ergonomia moderna ha profondamente trasformato i processi di progettazione e valutazione ergonomica, rendendoli più efficienti, flessibili e meno costosi. Tuttavia, nonostante questi progressi, rimane ancora una sfida colmare il divario tra ricerca e applicazione pratica, garantendo che queste tecnologie siano accessibili e scalabili nei contesti lavorativi reali.

1.3 Panoramica machine learning

Gli algoritmi di **ML (machine learning)**, un ramo dell'**AI (intelligenza artificiale)**, si basano sull'analisi dei dati per apprendere modelli e generare previsioni. A differenza dell'AI, che comprende qualsiasi sistema o macchina capace di eseguire compiti tipicamente associati all'intelligenza umana, il ML si concentra sullo sviluppo di algoritmi che permettono ai computer di apprendere dai dati ed effettuare previsioni senza essere esplicitamente programmati [22].

Gli algoritmi di **ML** si suddividono in tre categorie principali:

- **Apprendimento supervisionato**: processo in cui un **modello apprende a fare previsioni o classificazioni** sulla base di un insieme di **dati già etichettato**. In questo contesto, i dati di addestramento consistono in coppie di input e output, dove ogni input è associato a un'etichetta che rappresenta la risposta corretta [23].

- **Apprendimento non supervisionato**: processo che si avvale di **dati che non sono stati etichettati**. Il sistema deve quindi lavorare con le caratteristiche dei soli dati di input per scoprire strutture e pattern nascosti [23].

- **Apprendimento per rinforzo**: tecnica che punta a **realizzare agenti** autonomi in grado di scegliere azioni da compiere per il conseguimento di determinati obiettivi tramite **interazione con l'ambiente in cui sono immersi**. A differenza degli altri paradigmi, questo si occupa di problemi di decisioni sequenziali, in cui l'azione da compiere dipende dallo stato attuale del sistema e ne determina quello futuro [24].

Ognuna di queste categorie utilizza differenti tipologie di dati ed è progettata per obiettivi specifici [25]. Pur adottando metodologie diverse, questi approcci condividono due fasi essenziali. La prima è la **fase di training**, durante la quale il modello viene addestrato su un insieme di dati noti, apprendendo schemi e relazioni utili per elaborare previsioni. La seconda è la **fase di inferenza**, in cui il modello applica le conoscenze acquisite a nuovi dati mai visti prima, generando risposte o decisioni. Sebbene entrambe le fasi siano comuni a tutte le categorie di algoritmi, le modalità di implementazione variano in base all'approccio adottato.

1.3.1 Machine learning per l'ergonomia

L'**ML (machine learning)** nel settore **ergonomico industriale** viene impiegato principalmente per la **prevenzione primaria dei disturbi muscoloscheletrici (WMSD)**. Gli algoritmi di clustering non supervisionati vengono utilizzati per identificare e classificare i potenziali fattori di rischio associati ai WMSD. Inoltre, la combinazione di dispositivi indossabili e algoritmi di ML consente lo sviluppo di sistemi semi-automatizzati, in grado di monitorare in tempo reale il rischio di WMSD, migliorando la prevenzione e la sicurezza sul luogo di lavoro [26].

Un altro approccio prevede l'uso di **immagini RGB o RGB-D** acquisite da telecamere per stimare la cinematica 3D e il dispendio energetico meccanico durante le attività lavorative come mostrato in Figura 1.4. Questa metodologia permette di simulare gli output di strumenti biomeccanici tradizionali, riducendo così la dipendenza da attrezzature specializzate e costose [26].

Figura 1.4: Uso della visione artificiale per l'analisi ergonomica. I punti di riferimento articolari evidenziano un sistema di monitoraggio basato su machine learning per la valutazione del rischio ergonomico. [27].

I **sistemi basati su elettromiografia di superficie (sEMG)**, combinati con algoritmi di ML, possono rilevare automaticamente movimenti potenzialmente dannosi per i muscoli, specialmente durante operazioni di movimentazione dei materiali. Inoltre, algoritmi di ML come alberi decisionali, macchine a vettori di supporto (SVM), K-Nearest Neighbor (KNN) e random forest possono essere impiegati per classificare le **valutazioni del rischio** basate sull'equazione di sollevamento NIOSH, offrendo un supporto più preciso e automatizzato nella prevenzione dei disturbi muscoloscheletrici [28].

1.4 Approccio tecnologico

In questo contesto, il presente lavoro di progetto si propone di sviluppare un sistema per l'**addestramento di un manichino umanoide in ambiente virtuale**, con l'obiettivo di generare **movimenti naturali e realistici**.

Il sistema sarà realizzato attraverso l'integrazione di tecniche di intelligenza artificiale e simulazione fisica, al fine di ottenere risultati affidabili e applicabili a contesti reali. In particolare, l'adozione di modelli di **DRL (Deep Reinforcement Learning)** consentirà al manichino umanoide di apprendere in modo autonomo pattern di movimento coerenti con la dinamica umana, migliorando così l'aderenza ai principi biomeccanici.

Il flusso operativo principale inizia a partire dall'inserimento di un input testuale all'interno del game engine Unity, che viene successivamente elaborato dal modello text-to-motion pre-addestrato **MoMask** al fine di generare un'animazione da utilizzare come sequenza di riferimento per il comportamento motorio. L'animazione prodotta viene quindi importata nel sistema, opportunamente elaborata e utilizzata come base per l'addestramento dellagente AI.

L'agente ha il compito di apprendere il comportamento osservabile nell'animazione di riferimento e, contemporaneamente, portare a termine il task assegnato, che nel caso studio considerato consiste nel raggiungimento di una specifica posizione nello spazio. L'addestramento viene condotto utilizzando il framework **ML-Agents**, che consente di eseguire il ciclo di apprendimento attraverso l'algoritmo di ottimizzazione **Proximal Policy Optimization (PPO)**.

Durante il processo di addestramento, l'agente comunica all'algoritmo informazioni relative allo stato corrente della scena, sulla base delle quali vengono generate le azioni motorie, sotto forma di rotazioni e torsioni da applicare ai giunti target del manichino umanoide presente nell'ambiente simulato. Questo ciclo iterativo consente all'agente di affinare progressivamente il proprio comportamento, ottimizzando l'imitazione del movimento di riferimento e il raggiungimento del compito assegnato.

Il processo di apprendimento sarà monitorato tramite **TensorBoard**, uno strumento di analisi delle metriche di addestramento che permetterà di analizzare il comportamento dell'agente.

Capitolo 2

Tecnologie per la simulazione del movimento umano

L'**ergonomia negli ambienti industriali** si concentra sull'interazione tra l'uomo e il luogo di lavoro, con l'obiettivo di progettare postazioni sicure, efficienti e incentrate sull'uomo [20]. L'avvento dell'**Industria 4.0** ha introdotto nuove tecnologie, tra cui la VR (realtà virtuale) e la AR (realtà aumentata), che consentono di creare simulazioni accurate degli ambienti di lavoro fin dalle prime fasi di progettazione. Questo approccio, noto come ergonomia virtuale, sfrutta i **modelli umani digitali (DHM)** per simulare l'interazione tra l'operatore e la postazione di lavoro, permettendo di valutare la raggiungibilità dei punti di lavoro per diversi percentili antropometrici e di eseguire analisi dettagliate [20].

Per approfondire le tecnologie alla base della simulazione ergonomica e della riproduzione del movimento umano in ambienti virtuali, questo capitolo analizzerà i principali strumenti e metodologie impiegati nel settore. In particolare, verranno esaminati gli **ambienti di simulazione 3D**, valutando le loro caratteristiche e i **sistemi per la simulazione ergonomica**.

Successivamente, verranno esplorati gli approcci basati sull'AI (intelligenza artificiale) applicati all'ergonomia virtuale, con particolare attenzione ai sistemi di generazione del movimento umano e agli algoritmi di deep learning per l'addestramento del movimento. Infine, il capitolo si concluderà con una discussione sui criteri adottati per la scelta degli strumenti utilizzati in questo lavoro di ricerca, evidenziandone vantaggi e limitazioni rispetto ad altre alternative disponibili.

2.1 Introduzione alla realtà virtuale

Il termine **VR (realtà virtuale)** indica un insieme di tecnologie in grado di creare ambienti digitali che sostituiscono temporaneamente e in modo completo il mondo reale. Questa tecnologia si basa sulla capacità di generare esperienze immersive, coinvolgendo gli utenti in ambienti tridimensionali altamente realistici. [20]

La VR è caratterizzata da tre qualità fondamentali:

- **Immersione**, la capacità di percepire fisicamente di trovarsi nel mondo virtuale, influenzata da fattori come la risoluzione grafica e la latenza[29].
- **Presenza**, la sensazione soggettiva di appartenere al mondo virtuale, che inganna la mente facendola credere di vivere un'esperienza reale [29].
- **Interazione**, il grado in cui un utente può modificare l'ambiente virtuale in tempo reale [29].

Inizialmente sviluppata per il settore del **gaming**, la realtà virtuale ha progressivamente ampliato il proprio campo di applicazione, trovando impiego in molteplici ambiti. Nel settore **industriale**, viene utilizzata per ottimizzare le postazioni di lavoro, creare simulazioni virtuali, sviluppare prototipi digitali e migliorare la formazione del personale. Questa tecnologia consente di mantenere elevati standard di qualità e efficienza produttiva, contribuendo al tempo stesso al benessere e alla sicurezza dei lavoratori.

2.1.1 Tecnologie per la realtà virtuale

L'elevata capacità di immersione nella VR è resa possibile dalla creazione di ambienti ad alta definizione e con una visione a 360 gradi, oltre all'uso di dispositivi come visori guanti dotati di sensori e auricolari per veicolare l'esperienza ai sensi dell'utente, permettendo una completa immersione nella simulazione.

Attualmente, il mercato offre diverse **tecnologie** per la realizzazione di **ambienti virtuali**. Tra le configurazioni si trovano i sistemi CAVE (Cave Automatic Virtual Environment), che utilizzano da tre a sei schermi per circondare l'utente con proiezioni tridimensionali, come mostrato in Figura 2.1. Per fruire di questa esperienza, è necessario indossare occhiali 3D e interagire con l'ambiente attraverso un controller. Questo sistema considerato immersivo, offre una visione completa dello spazio virtuale e soprattutto risulta meno invasivo rispetto all'utilizzo di un visore VR. Tuttavia, la sua dimensione e il suo costo lo rendono uno strumento altamente specializzato, spesso impiegato in ambiti accademici, industriali e di ricerca, dove l'accuratezza e il livello di dettaglio dell'interazione virtuale sono prioritari rispetto alla portabilità e al costo.

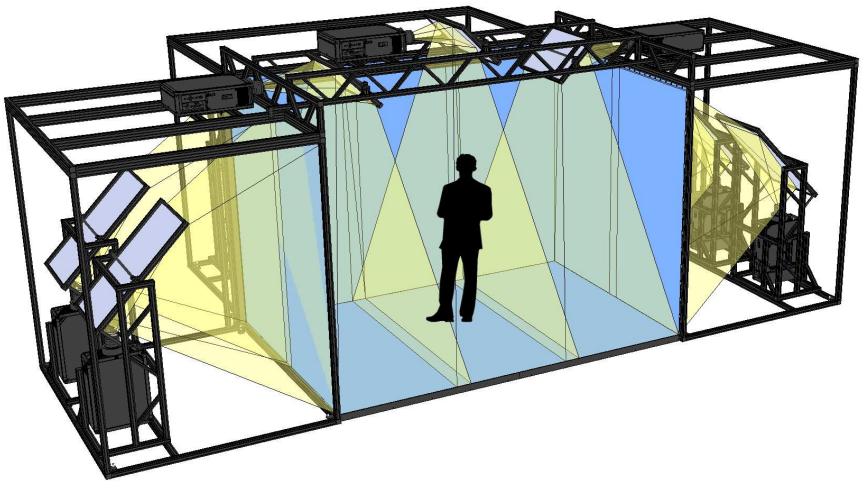


Figura 2.1: Rappresentazione di un sistema CAVE, un ambiente di realtà virtuale immersiva che utilizza proiettori per visualizzare immagini 3D su più superfici circostanti l'utente. Il sistema consente l'interazione con il mondo virtuale tramite occhiali 3D e controller, offrendo un'esperienza coinvolgente con minori effetti collaterali rispetto ai visori VR tradizionali. [30].

Tra le tecnologie più innovative e diffuse, figurano l'HTC VIVE [31] e l'Oculus [32]. Questi dispositivi sono progettati per offrire un'esperienza immersiva più avanzata grazie all'uso di un casco HMD (Head-Mounted Display) dotato di sensori di tracciamento del movimento. Gli utenti possono visualizzare immagini 3D stereoscopiche, determinare la propria posizione nello spazio virtuale e interagire con gli oggetti digitali mediante joystick. Inoltre, i visori sono spesso dotati di cuffie integrate, che permettono un'esperienza sonora ancora più coinvolgente. Rispetto ai sistemi CAVE, questi dispositivi sono più versatili e facilmente trasportabili, poiché non richiedono un'installazione complessa. Tuttavia, il loro utilizzo prolungato può provocare alcuni effetti collaterali, come mal di testa, nausea e affaticamento visivo.

2.1.2 Software per la creazione di esperienze VR

Per creare esperienze immersive in VR è necessario impiegare **software** avanzati in grado di gestire diversi aspetti, tra cui il rendering 3D, la fisica degli oggetti, le interazioni utente e l'integrazione con dispositivi VR come visori e controller. Tra le soluzioni più diffuse e versatili spiccano i motori di gioco **Unity** e **Unreal Engine**, che grazie alla loro compatibilità con le principali piattaforme VR e alla disponibilità di strumenti dedicati, rappresentano la scelta ideale per sviluppare ambienti immersivi.

Questi strumenti permettono di realizzare mondi virtuali realistici e interattivi, adattabili a diversi **ambiti applicativi**, dai videogiochi alla formazione, fino a settori come l'industria e la medicina.

Unity Engine

Unity è un motore di gioco multipiattaforma, sviluppato da Unity Technologies. Ampiamente utilizzato per la creazione di videogiochi, simulazioni interattive, esperienze di VR (realtà virtuale) e AR (realtà aumentata), risulta essere molto popolare per la sua versatilità, accessibilità e potenza [33].

Uno dei suoi **punti di forza** è la capacità multipiattaforma, che consente di sviluppare applicazioni per una vasta gamma di dispositivi. Grazie a un sistema grafico avanzato, supporta effetti visivi complessi, tra cui illuminazione globale, ombre dinamiche e rendering in tempo reale permettendo di creare ambienti immersivi e visivamente accattivanti [34].

Il motore integra strumenti avanzati per la fisica e l'animazione. La programmazione in Unity si basa su C#, un linguaggio potente e flessibile che consente di implementare logiche di gioco anche complesse. Inoltre, grazie all'Asset Store, gli sviluppatori hanno accesso a un vasto catalogo di modelli 3D, texture, script e altri asset predefiniti [34]. Inoltre, è supportato da una grande comunità di sviluppatori, che mette a disposizione forum, tutorial e risorse di apprendimento.

Unreal

Unreal Engine sviluppato da Epic Games, è uno dei motori di gioco più avanzati e potenti disponibili oggi. La sua straordinaria qualità grafica e la versatilità lo rendono ideale non solo per la creazione di videogiochi, ma anche per lo sviluppo di esperienze immersive VR e AR.

Unreal Engine è uno strumento con una grafica avanzata, che sfrutta tecnologie all'avanguardia per ottenere un elevato realismo. Tra queste, **Lumen**, un sistema di illuminazione globale dinamica, e **Nanite**, una tecnologia che consente il rendering in tempo reale di modelli estremamente dettagliati [35].

Il motore include strumenti di sviluppo avanzati, tra cui **Blueprint**, un sistema di scripting visivo che permette di creare logiche di gioco senza dover scrivere codice. Per chi invece necessita di un maggiore controllo, Unreal Engine supporta il linguaggio C++ [36].

Le capacità di animazione e simulazione fisica sono un altro **punto di forza** del motore. Strumenti avanzati consentono di animare personaggi con un livello di dettaglio impressionante, mentre il sistema basato sull'intelligenza artificiale, noto come **Taipei**, migliora la fluidità dei movimenti e la reattività delle animazioni.

Tuttavia, per sfruttare appieno le potenzialità di Unreal Engine, è necessario disporre di un **hardware potente**. Infatti, il motore richiede schede grafiche di fascia alta, come le NVIDIA RTX o le AMD Radeon RX 5700 XT, per garantire prestazioni ottimali [37].

Confronto tra Unity e Unreal Engine

I due motori di gioco appena presentati sono fra i più diffusi al mondo, ciascuno con **caratteristiche** uniche che lo rendono più adatto a determinati progetti rispetto ad altri. In particolare i punti che distinguono questi due sistemi sono:

- **Facilità di apprendimento**; Unity è più accessibile grazie all'interfaccia intuitiva e all'uso di C#, mentre Unreal Engine ha una curva di apprendimento più ripida, soprattutto per chi non ha esperienza con C++.
- **Supporto cross-platform**: Unity offre un supporto multipiattaforma più ampio, specialmente per mobile e VR, con ottimizzazioni efficienti.
- **Grafica e rendering**: Unreal Engine è superiore per qualità visiva, grazie a tecnologie come Lumen e Nanite. Unity, pur offrendo buoni strumenti grafici, ha limiti nel rendering avanzato.
- **Prestazioni**: Unreal Engine è più ottimizzato per gestire scene complesse e giochi di grandi dimensioni, ma richiede hardware potente. Unity è più leggero e scalabile, ma può soffrire di problemi di prestazioni in progetti molto grandi.

Nel contesto dello **sviluppo di questo progetto**, il motore di simulazione Unity è stato ritenuto la soluzione più adatta. Sebbene non rappresenti lo standard di riferimento in termini di prestazioni grafiche o ottimizzazione del rendering, Unity si distingue per la sua flessibilità, la facilità d'uso, che consente di sviluppare applicazioni interattive in modo relativamente rapido e modulare.

Un ulteriore fattore determinante nella scelta è stato il forte supporto della community, che offre documentazione estesa. Inoltre, la continuità con progetti preesistenti ha facilitato l'integrazione e il riutilizzo di componenti, riducendo i tempi di sviluppo e aumentando la coerenza architetturale dell'ambiente software.

2.2 Modelli umani digitali (DHM)

I **Digital Human Models (DHM)** sono **rappresentazioni tridimensionali di esseri umani** all'interno di ambienti virtuali e vengono impiegati per analizzare la sicurezza e le prestazioni di un progetto prima della realizzazione di un prototipo fisico [38].

L'uso di questi modelli digitali si è diffuso in diversi settori grazie alla loro capacità di migliorare la progettazione e l'organizzazione dei processi lavorativi. Nell'ambito della formazione del personale, l'impiego dei DHM consente di creare simulazioni interattive che rendono l'apprendimento più efficace, riducendo al contempo i costi legati all'insegnamento [39]. In campo ergonomico, le simulazioni predittive permettono di analizzare e prevenire l'insorgenza di disturbi muscoloscheletrici, favorendo una maggiore produttività e un miglior benessere per i lavoratori. Anche il settore sanitario trae notevoli benefici dall'utilizzo di questi strumenti, soprattutto per quanto riguarda la sicurezza nelle operazioni laparoscopiche [40], la progettazione di protesi personalizzate [41] e lo sviluppo di sistemi di assistenza avanzati per persone con disabilità [42].

2.2.1 Software per la simulazione ergonomica

Per la simulazione ergonomica sono stati sviluppati numerosi **software** specifici, ognuno con caratteristiche peculiari. **RAMSIS**, un sistema tedesco impiegato principalmente nel settore automobilistico, è uno dei più avanzati nel prevedere la postura e il comportamento degli occupanti all'interno dei veicoli, offrendo un'analisi dettagliata della loro interazione con l'abitacolo [43, 44]. **Jack**, software inizialmente progettato per la NASA e oggi integrato nei sistemi Siemens, è uno strumento particolarmente utile nelle simulazioni ergonomiche grazie alla possibilità di personalizzare i manichini digitali e di calcolare le coppie articolari necessarie per eseguire determinati movimenti [45, 46, 47]. Infine, **Human Model**, sviluppato da Fiat Chrysler Automobiles (FCA), si distingue dagli altri per la sua implementazione in Microsoft Excel, offrendo una soluzione rapida ed efficace per valutazioni ergonomiche, particolarmente utile per coloro che non hanno una formazione specialistica nell'ambito dell'ergonomia [48]. Questi software di simulazione ergonomica non utilizzano modelli di intelligenza artificiale basati su machine learning o deep learning, ma si basano su modelli antropometrici predefiniti, biomeccanica classica e algoritmi di ottimizzazione per prevedere la postura e il movimento umano.

2.3 Analisi Ergonomica tramite AI

Le nuove tecnologie incentrate sull'AI (intelligenza artificiale), svolgono un ruolo chiave nell'**ergonomia virtuale**. Il loro impiego migliora la progettazione degli

ambienti lavorativi e permette una valutazione dei rischi ergonomici più precisa, veloce ed efficace. Un esempio significativo è rappresentato dalla **predizione posturale**, ottenuta grazie all'uso di Artificial Neural Network (ANN), che consentono di prevedere le posture del corpo umano in diverse condizioni lavorative [49, 50, 51]. Nel campo dell'analisi della postura umana, l'AI trova applicazione nel **riconoscimento di modelli di movimento** durante le operazioni di assemblaggio manuale, così come nell'analisi del rischio ergonomico, dove, attraverso l'uso di immagini di profondità, è possibile calcolare in tempo reale il punteggio ergonomico e fornire un feedback immediato [52]. Un altro ambito di applicazione è l'**automazione delle valutazioni ergonomiche**, in cui le Convolutional Neural Network (CNN) permettono di eseguire analisi ergonomiche automatiche basate su immagini RGB-D, aumentando l'accessibilità e l'efficienza del processo [26].

Nel settore della formazione e della valutazione delle postazioni di lavoro, l'AI viene impiegata per la **personalizzazione degli ambienti di lavoro**, adattando processi e spazi lavorativi alle caratteristiche dei lavoratori [53]. In aggiunta, grazie la combinazione tra VR e AI è possibile creare **ambienti di formazione immersivi**, migliorando l'apprendimento e la consapevolezza ergonomica [54].

Un'area di grande interesse per gli studi ergonomici è la **simulazione del movimento umano basata su AI**. Questo approccio non solo si applica a molte delle aree precedentemente menzionate, ma offre anche significativi vantaggi in termini di riduzione dei tempi e dei costi, migliorando al contempo la precisione e l'obiettività delle valutazioni ergonomiche [20]. Le tecnologie utilizzate per la generazione e simulazione dei movimenti umani, includono tecniche avanzate per modellare e prevedere il movimento in contesti ergonomici e industriali. I diversi approcci applicativi prevedono:

- **Recurrent Neural Network (RNN)** e le sue varianti, per modellare dati sequenziali e prevedere i movimenti futuri in base agli schemi appresi [55].
- **Convolutional Neural Network (CNN)** e **Graph Convolutional Network (GCN)** che consentono di catturare le dipendenze spaziali nel movimento umano, analizzando le correlazioni tra le articolazioni per una rappresentazione più accurata [55].
- **Generative Adversarial Networks (GAN)**, impiegate nella generazione di dati sintetici, utilizzate per la previsione probabilistica del movimento, migliorando la qualità delle simulazioni attraverso un processo di apprendimento competitivo [55].
- **Phase-Functioned Neural Networks** per apprendere un ampio repertorio di movimenti, garantendo una maggiore adattabilità alle condizioni ambientali diversificate [56, 57].

- **DRL (Deep Reinforcement Learning)** per la definizione di agenti per l'apprendimento e l'esecuzione di varie attività attraverso tentativi ed errori, riducendo la necessità di intervento umano [58, 59]. Approccio particolarmente utile per controllare personaggi in ambienti basati sulla fisica, imitando clip di movimento di esempio e adattandosi a variazioni morfologiche [57, 56].

- **Macchina a stati neurale**, utilizzata per modellare movimenti sia periodici che aperiodici, incorporando internamente una macchina a stati per il controllo avanzato del personaggio, migliorandone la coerenza e la fluidità [56].

Questi approcci possono essere combinati e adattati in base all'applicazione specifica per ottenere risultati ottimali. Tuttavia, l'uso dell'AI per la simulazione del movimento umano in ergonomia presenta alcune sfide che devono essere affrontate per garantirne un'applicazione efficace. Una delle principali difficoltà riguarda la complessità dell'implementazione, che richiede competenze avanzate in ML e modellazione umana, oltre a una profonda comprensione delle tecniche di simulazione [20]. Un ulteriore aspetto fondamentale è la necessità di dati di alta qualità per l'addestramento dei modelli, un requisito essenziale per garantire previsioni precise e affidabili [54]. Nel contesto della generazione e dello studio del movimento umano, i dataset più utilizzati sono quelli provenienti da sistemi di Motion Capture (mocap), in quanto forniscono una rappresentazione dettagliata e coerente dei movimenti, permettendo di modellare con maggiore fedeltà le dinamiche corporee.

2.3.1 Dataset di movimenti umani

I dataset di **motion capture** rappresentano una risorsa fondamentale per la ricerca. Grazie a queste raccolte di dati, è possibile studiare il **movimento umano** in dettaglio, sviluppare modelli predittivi e migliorare la qualità delle simulazioni digitali. Fra i dataset più grandi troviamo:

- **AMASS (Archive of Motion Capture as Surface Shapes)**, che raccoglie e unifica ben 15 diversi dataset di motion capture ottico basati su marker. Grazie a un modello comune basato sul modello **SMPL**, AMASS offre oltre 40 ore di dati di movimento coinvolgendo più di 300 soggetti e registrando più di 11.000 movimenti. Ogni frame di questo dataset include informazioni dettagliate sulle forme 3D del corpo, sulle caratteristiche dei tessuti molli e sui parametri di posa, offrendo una rappresentazione estremamente dettagliata del movimento umano. Inoltre, AMASS non si limita a fornire dati in forma di scheletri e marker, ma mette a disposizione anche mesh 3D completamente rigged, rendendolo particolarmente utile per applicazioni avanzate come la grafica computerizzata e la simulazione biomeccanica [60].

- **Human3.6M**, che rappresenta uno dei più grandi dataset di pose umane 3D ad alta precisione. Questo archivio raccoglie 3.6 milioni di pose, acquisite da 11 soggetti (5 donne e 6 uomini), registrati da 4 diverse angolazioni. Ciò che lo caratterizza è la presenza di dati sincronizzati tra immagini 2D e 3D, motion capture e scansioni time-of-flight, oltre a dettagliate scansioni corporee 3D di tutti i soggetti coinvolti. Il dataset è organizzato in 15 scenari di training [61].

Oltre ai grandi archivi, esistono dataset più specifici che si concentrano su particolari tipologie di movimento o applicazioni. **MPI-HDM05** e **MPI-Pose Limits**, ad esempio, contengono un numero più ridotto di sequenze, rispettivamente 215 e 35 movimenti, ma offrono dati molto precisi per studi dettagliati sulla postura e sulla biomeccanica. Altri dataset, come **KIT**, **BMLrub** e **ACCAD**, raccolgono migliaia di movimenti eseguiti da decine di soggetti, garantendo una vasta gamma di variazioni interindividuali [60].

Un ruolo chiave nella ricerca è svolto anche dai dataset che combinano dati motion capture con immagini video sincronizzate, come **HumanEva**. Questo archivio fornisce circa 40.000 frame di motion capture sincronizzati con video multi-view, offrendo una risorsa preziosa per la valutazione di algoritmi di computer vision e riconoscimento del movimento. Analogamente, il dataset **TotalCapture** raccoglie dati multi-sensore, utili per analisi approfondite [62].

Un altro dataset di grande rilevanza è **NTU-RGB-D**, considerato il più grande archivio di movimento umano disponibile. Inizialmente conteneva oltre 100.000 movimenti distribuiti in 120 classi di azioni, ma è stato successivamente riorganizzato per migliorare l'accuratezza delle annotazioni. Tuttavia, i dati di pose provenienti dal sensore MS Kinect sono stati considerati rumorosi e instabili nel tempo, rendendo necessaria una ri-annotazione più precisa per migliorare la qualità delle sequenze [63].

In ambito medico e biomeccanico, dataset come **HumanAct12** e **EKUT** forniscono dati di alta qualità per l'analisi del movimento umano in contesti riabilitativi. Ad esempio, **HumanAct12** offre 1.191 clip di movimento suddivise in 12 categorie di azioni e 34 sottocategorie, fornendo annotazioni dettagliate delle posizioni 3D dei giunti [63, 60]. Allo stesso modo, dataset come **Transitions** e **SSM (Synchronized Scans and Markers)** combinano dati motion capture con scansioni 3D sincronizzate, offrendo una visione ancora più dettagliata del movimento umano.

Modelli SMPL

Per **rappresentare efficacemente i dati** di movimento umano nei contesti di simulazione e analisi ergonomica, è fondamentale adottare **modelli antropometrici** accurati. In questo contesto, i modelli SMPL offrono una rappresentazione

parametrica del corpo umano che permette di ricostruire e manipolare i movimenti estratti dai dataset.

Il modello SMPL (Skinned Multi-Person Linear Model) è una rappresentazione del corpo umano basato sui vertici, progettato per essere compatibile con i software e i motori di rendering grafici esistenti. I **vantaggi** principali che presenta questo modello sono il realismo, in quanto permette di rappresentare accuratamente una vasta gamma di forme del corpo umano in pose naturali. Inoltre, è efficiente da animare e può essere utilizzato per generare animazioni in tempo reale [64]. La sua struttura è definita da uno scheletro articolato a cui è associata una mesh definita da vertici, le cui deformazioni sono regolate da blend shape correttivi appresi da dati reali di scansioni 3D. Una delle caratteristiche principali di SMPL è la sua capacità di modellare non solo i diversi percentili tra individui, ma anche le deformazioni che si verificano quando il corpo è in movimento o assume diverse pose. Queste deformazioni dipendono sia dalla struttura scheletrica che dalla posa. La Figura 2.2 mostra la gerarchia e la struttura del modello con parametri di forma differenti.

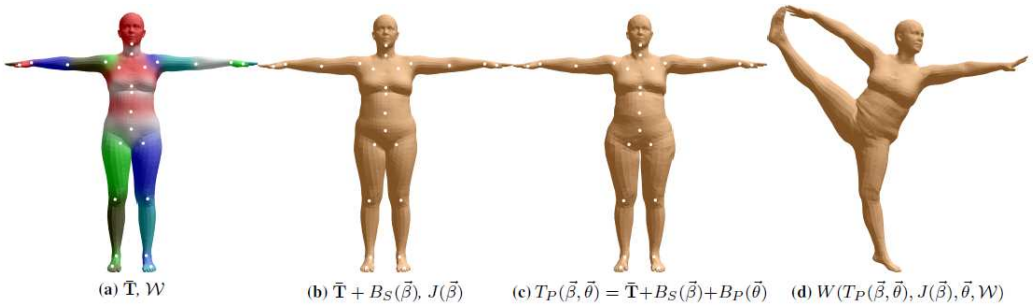


Figura 2.2: (a) Template mesh con pesi blend indicati dai colori e articolazioni mostrate in bianco. (b) Mesh con contributo di blendshape guidato dall'identità, rappresentato dal vettore di forma $\vec{\beta}$. Le posizioni dei vertici e delle articolazioni variano in modo lineare in funzione del vettore di forma. (c) Mesh con l'aggiunta di blendshape dipendenti dalla posa $\vec{\theta}$. Si nota un'espansione delle anche rispetto alla versione precedente, suggerendo l'influenza delle deformazioni dipendenti dalla posa. (d) Vertici deformati riposizionati tramite skinning con dual quaternion per la posa finale.

Per una rappresentazione ottimale delle deformazioni dipendenti dalla posa, SMPL utilizza una funzione lineare delle matrici di rotazione delle articolazioni. Questo approccio permette di ottenere un modello che è contemporaneamente accurato, semplice da implementare e altamente efficiente dal punto di vista computazionale, rendendolo adatto per applicazioni che richiedono la riproduzione di animazioni in tempo reale. Come descritto precedentemente, il comportamento

del modello SMPL è basato su dataset di scansioni 3D [65] di corpi umani in varie pose. Per il processo di apprendimento sono state utilizzate tecniche di apprendimento statico, per addestrare i parametri delle blend shape correttive (che variano tra individui) e delle deformazioni dipendenti dalla posa (che variano in base al movimento).

Diversi sono i **campi di utilizzo** del modello SMPL. Essi vanno dalla produzione di film e videogiochi alla simulazione ergonomica. La sua compatibilità con software come Unity, Blender e Maya lo rende uno strumento versatile.

Biovision hierarchy file

Per utilizzare efficacemente i dataset di movimenti umani nei modelli SMPL, è necessario disporre di formati standardizzati che rappresentino con precisione la cinematica articolare. Tra questi, i file **BVH (Biovision Hierarchy)** sono ampiamente utilizzati per la registrazione e la **trasmissione di dati di movimento**. Questa tipologia di documenti contiene sia le informazioni delle ossa, come descritte in Figura 2.3, che compongono lo scheletro, sia i dati dell'animazione in un unico file [66].

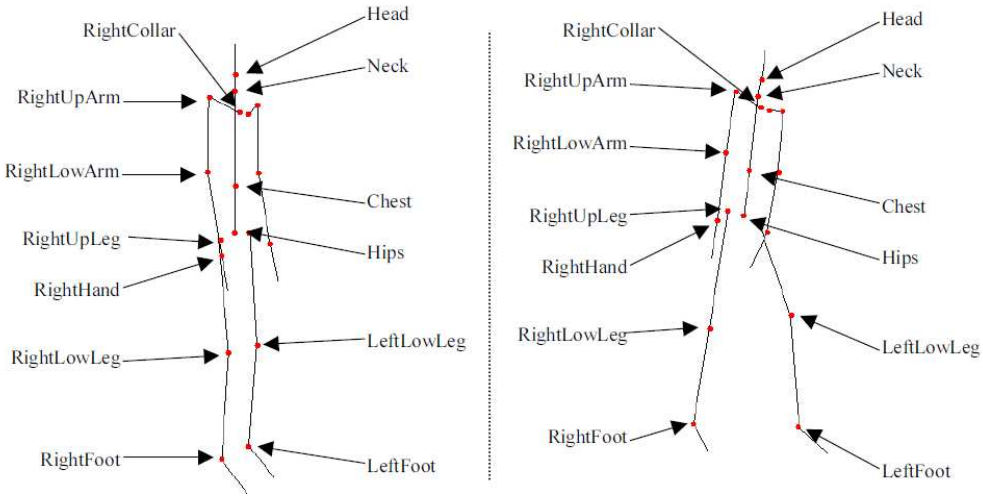


Figura 2.3: Struttura scheletrica di un semplice file BVH.

Il formato è suddiviso in due sezioni; la prima definisce la gerarchia e la posizione iniziale dello scheletro a riposo. Mentre, la seconda descrive i dati del canale per ogni frame, definendo quindi il movimento. I dati vengono rappresentati attraverso un formato testuale che fa uso di parentesi graffe per la suddivisione delle sezioni dello scheletro. La sezione dedicata alla gerarchia inizia con il keyword *HIERARCHY*, seguito dalla definizione del *ROOT* del modello scheletrico. Ogni osso è definito da un nodo della gerarchia e contiene le seguenti informazioni:

- Il nome dell'osso.
- L'attributo *OFFSET* indicare la traslazione rispetto al suo nodo padre.
- La linea *CHANNELS* specificare i gradi di libertà.

La struttura scheletrica viene descritta in maniera ricorsiva, includendo figli e endpoint come *JOINT* ed *End Site*. Conclusa la gerarchia, segue la sezione *MOTION*. Essa contiene il numero totale di frame e la durata di ciascun di essi (Frame Time). In più i dati di movimento definiscono la traslazione rispetto l'origine e la rotazione per ogni giunto in ogni frame, seguendo l'ordine specificato nella gerarchia.

2.3.2 Generazione di movimenti umani

I modelli di generazione di movimenti umani hanno l'obiettivo di creare sequenze di animazioni 3D di figure umane in base a diversi input, come testo, azioni o sequenze di movimento precedenti [67]. Negli ultimi anni, la ricerca ha individuato cinque principali categorie di modelli per affrontare questo compito:

- **Modelli autoregressivi con quantizzazione vettoriale**: questi sistemi trasformano il movimento in sequenze di token discreti e li generano unidirezionalmente. Un esempio è **T2M-GPT**, che utilizza un'architettura GPT per predire le sequenze di movimento. Sebbene questo approccio sia relativamente semplice ed efficace, la quantizzazione può introdurre errori e limitare l'espressività del movimento [68].
- **Modelli generativi mascherati**: ispirati ai transformer come **BERT**, questi modelli prevedono iterativamente le parti di movimento mancanti [68]. Il più avanzato è **MoMask**, che grazie a una quantizzazione gerarchica e a un doppio modello transformer riesce a generare animazioni fluide e realistiche, superando le limitazioni dei modelli autoregressivi e di diffusione [67].
- **Modelli di diffusione**: questi sistemi creano movimenti progressivamente, rimuovendo il rumore da una sequenza iniziale casuale. Esempi noti sono **MotionDiffuse** e **MDM (Motion Diffusion Model)** [67, 69]. Sebbene questi modelli garantiscano una grande diversità di movimenti, sono computazionalmente costosi e richiedono numerose iterazioni per ottenere risultati di qualità [68].
- **Autoencoder variazionali (VAE)**: questi modelli imparano una rappresentazione compatta del movimento e generano nuove sequenze campionando dallo spazio latente. Metodi come **TEMOS** e **ACTOR** risultano particolarmente

utili per la sintesi di movimenti realistici, ma possono offrire meno varietà rispetto ai modelli di diffusione [67].

- **Approcci a due stadi**: separano la generazione del movimento in due fasi, prima stimando la lunghezza dell'animazione e poi generando la sequenza corrispondente. Sebbene permettano un maggiore controllo temporale, introducono una maggiore complessità nel processo di apprendimento [70].

Dal confronto tra questi modelli emerge come **MoMask** riesca a garantire animazioni di qualità superiore rispetto a **MDM** e **T2M-GPT**, con tempi di inferenza più rapidi rispetto ai modelli di diffusione. Inoltre, assicura una maggiore coerenza tra il testo di input e il movimento generato, ottenendo risultati eccellenti su dataset standard come **HumanML3D** e **KIT-ML** [68]. Grazie alla sua struttura, MoMask rappresenta ad oggi la soluzione migliore per la generazione automatica di movimenti umani realistici da testo.

2.3.3 Generazione di movimenti umani in scene 3D

Negli ultimi anni, la generazione di movimenti umani in ambienti tridimensionali ha suscitato un interesse crescente, spingendo la ricerca verso lo sviluppo di tecniche sempre più sofisticate. Se da un lato i modelli di **text-to-motion** permettono di creare animazioni basate su descrizioni testuali, la loro principale limitazione è l'assenza di vincoli fisici, come la gravità e le interazioni con l'ambiente. Questo ha portato a un'evoluzione degli approcci esistenti, con l'obiettivo di produrre movimenti più realistici, capaci di adattarsi a scenari complessi in cui forze, collisioni e dinamiche ambientali giocano un ruolo fondamentale.

Un'importante direzione di ricerca riguarda lo sviluppo di politiche di controllo, ossia strategie computazionali che consentono a personaggi virtuali di muoversi in modo realistico all'interno di ambienti 3D. Per raggiungere questo obiettivo, molti studi si basano su tecniche avanzate di **Deep Reinforcement Learning**, permettendo agli agenti virtuali di apprendere comportamenti sempre più adattivi e naturali. Alcuni algoritmi sfruttano reti neurali per sviluppare una politica di controllo multi-obiettivo, apprendendo movimenti a partire da dati non strutturati [71].

Un'altra strategia particolarmente efficace è la **sintesi di controller**, ovvero la creazione di algoritmi che consentono ai personaggi animati di riprodurre movimenti di riferimento cinematici mantenendo, al contempo, il rispetto degli obiettivi del compito assegnato [72].

Parallelamente, alcuni studi si concentrano sulla creazione di agenti pre-addestrati in grado di affrontare una vasta gamma di compiti in ambienti complessi. L'idea alla base di questi approcci è lo sviluppo di modelli di fondazione comportamentali

per agenti umanoidi, capaci di eseguire movimenti articolati grazie a un controllo avanzato dell'intero corpo, basato su osservazioni propriocettive [73].

Un ulteriore approccio innovativo nella simulazione dei movimenti umani combina tre elementi chiave: **comprensione del linguaggio naturale, rappresentazione spaziale e modellazione generativa**. In questo contesto, si utilizzano grandi modelli linguistici (LLM) per interpretare istruzioni testuali, sensori volumetrici per ricostruire l'ambiente tridimensionale e modelli di diffusione per generare movimenti coerenti con il contesto circostante. Un aspetto particolarmente interessante di questa tecnica è la possibilità di modellare le interazioni tra l'umanoide e gli oggetti presenti nella scena, creando animazioni più credibili e immersive. Tuttavia, anziché affidarsi a un motore fisico tradizionale, l'ambiente virtuale adotta un sistema che simula il comportamento fisico senza basarsi su dinamiche rigidamente predefinite [74].

Nei paragrafi successivi verranno analizzati nel dettaglio questi modelli, mettendo in evidenza le loro caratteristiche, i vantaggi e le potenziali applicazioni nei diversi ambiti della simulazione tridimensionale.

MetaMotivo

MetaMotivo rappresenta un **innovativo modello comportamentale di base per umanoidi**, progettato per affrontare compiti di controllo di tutto il corpo in modalità zero-shot. Questo significa che, dopo un unico processo di addestramento su un ampio set di dati non etichettati, il modello è in grado di generalizzare a una varietà di compiti senza richiedere un ulteriore affinamento specifico per ciascuna attività [73].

L'addestramento si basa su un nuovo algoritmo di reinforcement learning non supervisionato, denominato **FB-CPR (Forward-Backward Representations with Conditional Policy Regularization)**. Il principio chiave di FB-CPR è l'integrazione di rappresentazioni forward-backward, che consentono di strutturare traiettorie non etichettate, stati, ricompense e politiche all'interno dello stesso spazio latente. Inoltre, un discriminatore condizionato al latente guida il modello nell'apprendimento di politiche che coprano efficacemente gli stati osservati nel dataset comportamentale, migliorando così la generalizzazione [73].

Diverse sono le innovazioni introdotte da MetaMotivo che lo rendono un modello altamente promettente. La generalizzazione Zero-Shot permette di affrontare una vasta gamma di compiti di controllo motorio senza necessità di un addestramento specifico per ogni attività. In più presenta la generazione di **comportamenti naturali e umani** e un **bias umano intrinseco** che porta il modello a risolvere i compiti in modo intuitivo, trovando soluzioni che ricordano il comportamento umano e risultano più naturali rispetto a quelle ottenute con agenti RL tradizionali[73].

Nonostante i suoi numerosi vantaggi, MetaMotivo presenta alcune criticità, che si possono riassumere in un divario prestazionale rispetto a soluzioni ottimizzate per singoli compiti, rumore e ertefatti nei dati di motion capture e una carenza dello studio e valutazione del movimento umano, che rimangono essenziali per validazione del modello [73].

DeepMimic

DeepMimic è un framework per l'animazione di personaggi virtuali che combina l'apprendimento per rinforzo guidato da obiettivi con l'utilizzo di dati di riferimento, come clip di motion capture o animazioni keyframe. L'obiettivo principale è la **sintesi di un controller** che consenta a un personaggio simulato di imitare movimenti di riferimento, mantenendo al contempo la capacità di soddisfare obiettivi specifici del compito assegnato [72].

L'architettura di DeepMimic si basa su un apprendimento per rinforzo guidato da esempi, in cui una policy di controllo viene addestrata utilizzando l'**algoritmo Proximal Policy Optimization (PPO)**. La funzione di ricompensa include un termine che incentiva il personaggio simulato a imitare un movimento di riferimento e un altro che lo spinge a raggiungere obiettivi specifici del compito [72].

Il sistema si distingue per la sua capacità di apprendere e riprodurre un'ampia gamma di movimenti, rendendolo un modello estremamente versatile. Un aspetto particolarmente rilevante è la possibilità di combinare l'imitazione dei movimenti di riferimento con il raggiungimento di obiettivi specifici, garantendo un equilibrio tra fedeltà all'animazione originale e adattabilità alle esigenze del compito. Inoltre, le politiche di controllo sviluppate risultano compatte ed efficienti dal punto di vista computazionale, contribuendo a ridurre il costo dell'elaborazione. Anche in questo caso il modello ha la caratteristica di apprendere da dati di motion capture non organizzati, senza richiedere una rigorosa etichettatura [72].

Nonostante i vantaggi, il framework presenta alcune limitazioni. In presenza di perturbazioni impreviste, i movimenti del personaggio possono apparire rigidi e meno reattivi. Inoltre, il modello deve trovare un equilibrio tra la fedeltà all'imitazione e l'efficacia nel raggiungere gli obiettivi del compito. Infine, l'addestramento delle reti neurali utilizzate da DeepMimic, inoltre, richiede ingenti risorse computazionali e lunghi tempi di elaborazione, rendendo il processo particolarmente oneroso [72].

2.4 Sistemi e Strumenti per l'implementazione

La scelta delle tecnologie e degli strumenti adottati in questo lavoro è stata orientata dallo scopo principale di sviluppare un sistema di addestramento per un manichino umanoide in ambiente virtuale, capace di generare movimenti naturali e realistici attraverso l'integrazione di intelligenza artificiale e simulazione fisica.

Per la realizzazione della simulazione e l'addestramento, è stata selezionata la piattaforma di Unity per la sua compatibilità con i progetti esistenti e per la sua accessibilità rispetto ad alternative come Unreal Engine. Inoltre, grazie al framework **ML-Agents**, è possibile integrare direttamente algoritmi di **apprendimento per rinforzo profondo**, semplificando l'implementazione e il processo di addestramento.

Per garantire **animazioni di riferimento precise e adattabili**, il sistema sfrutta **MoMask**, un modello basato su reti neurali per la generazione di movimenti a partire da descrizioni testuali. La scelta di questo approccio è stata motivata dalla volontà di rendere il sistema accessibile e facilmente utilizzabile, permettendo la generazione di animazioni in modo semplice e intuitivo, senza la necessità di competenze tecniche avanzate. Tale impostazione ha rappresentato un compromesso efficace tra la qualità del movimento e la praticità nella raccolta dei dati, in quanto ha ridotto la necessità di interventi manuali nella fase di validazione del dataset e di selezione delle animazioni più idonee. In questo modo è stato possibile ottimizzare le risorse disponibili, delegando le fasi più onerose della raccolta dati al sistema automatizzato e concentrando maggiore attenzione e tempo sulle successive fasi di sviluppo e implementazione del progetto.

Il lavoro è stato avviato a partire da una base preesistente realizzata all'interno del motore Unity [75], basata su un modello di esempio fornito dal framework ML-Agents [76] e rappresentante un manichino umanoide. In questa implementazione iniziale erano già definite le policy di interazione tattile tra il manichino e l'ambiente simulato, nonché la logica di gestione dei giunti articolari, includendo il controllo delle rotazioni e delle forze applicate. A partire da questa architettura, il lavoro è stato esteso integrando i principi e il comportamento logico ispirata al progetto DeepMimic [72], con l'obiettivo di implementare una policy di apprendimento per imitazione più avanzata rispetto quella del progetto di partenza.

L'intero sistema segue un **flusso operativo iterativo**, in cui le animazioni generate da **MoMask** vengono utilizzate come riferimento per l'addestramento dell'agente. Quest'ultimo, interagendo con la fisica del motore di simulazione, apprende progressivamente a eseguire i movimenti con maggiore precisione e coerenza. Nei paragrafi successivi verranno illustrati nel dettaglio gli strumenti e le metodologie impiegate in questo lavoro [72].

2.4.1 MoMask

MoMask è un nuovo framework di modellazione generativa, che permette di creare movimenti umani 3D passando come **input** un **prompt testuale** e restituendo come **output** un file **BVH** che descrive l'animazione generata. Il modello si basa su uno **schema di quantizzazione gerarchico** per rappresentare il movimento umano come **token di movimenti discreti multistrato** con dettagli ad alta fedeltà. L'architettura del modello descritta in Figura 2.4, comprende:

- Un modello di apprendimento automatico a **quantizzazione residua basata sul movimento (VQ-VAE)** che combina un **autoencoder variazionale (VAE)** con la **quantizzazione vettoriale (VQ)**. In particolare, questo modello permette di **convertire** i **movimenti 3D in token discreti multistrato**. Ciò significa che ogni movimento viene scomposto in una serie di azioni più semplici che rappresentano diversi livelli di dettaglio del movimento. La struttura del modello prevede che i livelli più bassi rappresentano la parte principale del movimento, mentre i livelli superiori aggiungono dettagli più fini.

- Un modulo **masked transformer (M-Transformer)** che inizializza una sequenza vuota di movimento e, partendo dalle descrizioni testuali, completa progressivamente le parti mancanti.

- Un **residual transformer (R-Transformer)** che dopo l'elaborazione del M-Transformer, aggiunge dettagli residui all'animazione, rendendo i movimenti più realistici e precisi.

Il modello è in grado di generare una sequenza di movimento 3D a partire da una descrizione testuale. In particolare, questo processo è suddiviso in tre fasi. Nella prima avviene la **generazione della sequenza di base**, ovvero una sequenza dove tutti i token che descrivono il movimento sono nascosti.

Utilizzando la sequenza vuota, M-Transformer comincia a **riempire i token mascherati in modo iterativo**, dove ad ogni iterazione, cerca di prevedere quali parti del movimento dovrebbero andare nei token mascherati.

Le parti che il modello predice con **confidenza maggiore vengono confermate**, mentre quelle di cui c'è **maggiore incertezza vengono nuovamente mascherate** per essere predette nuovamente. Il processo appena descritto prende il nome di **remasking** e continua fino a quando tutta la sequenza base non viene completata. Infine, vengono **aggiunti i dettagli più fini alla sequenza**, utilizzando la R-Transformer. Partendo dalla sequenza base, il modello aggiunge i dettagli fini all'animazione, ed ogni livello di dettaglio si basa sui livelli precedenti. In questo modo, ogni nuova aggiunta di informazioni tiene conto di quello che è già stato generato nel livello precedente.

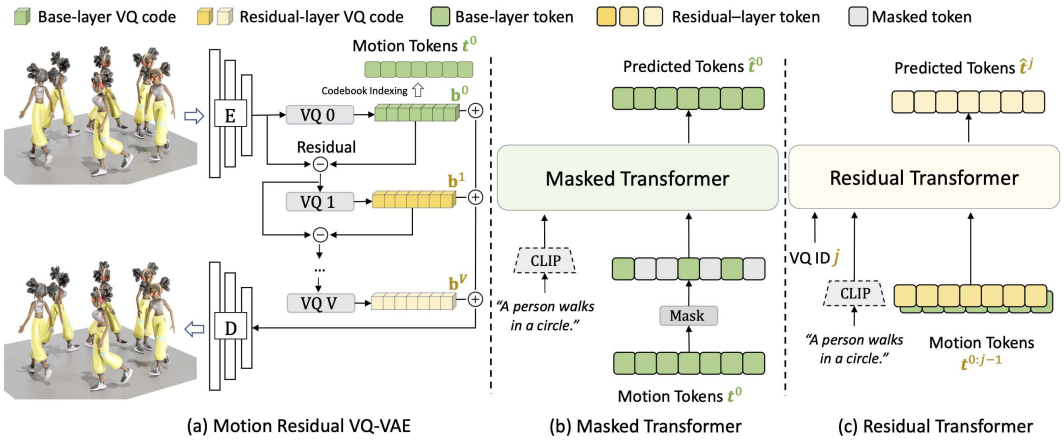


Figura 2.4: (a) **Tokenizzazione della sequenza di movimenti**: La sequenza di movimenti viene trasformata in token discreti tramite la quantizzazione vettoriale (VQ), chiamata anche strato di quantizzazione di base. Questo processo viene applicato anche in modo gerarchico su più livelli per la quantizzazione residua. (b)**Predizione parallela tramite M-Transformer**: Nel livello di base t^0, i token vengono mascherati casualmente. Un M-Transformer condizionato dal testo viene quindi addestrato per prevedere i token mascherati nella sequenza contemporaneamente. (c) **Predizione progressiva strato per strato tramite il R-Transformer**: Tramite R-Transformer condizionato dal testo impara a prevedere progressivamente i token residui $t^{j>0}$ a partire dai token nei livelli precedenti $t^{0:j-1}$. Questo permette un raffinamento progressivo dei movimenti previsti.[68]

Per estrarre l'animazione, si passa all'ultima fase di **decodifica** che include tutti i token generati (quelli della sequenza base e quelli dei livelli di dettaglio). In pratica, i vari token vengono tradotti in una sequenza continua di pose 3D, utilizzando il decodificatore della VQ-VAE. L'intero processo di inferenza è illustrato nella Figura 2.5.

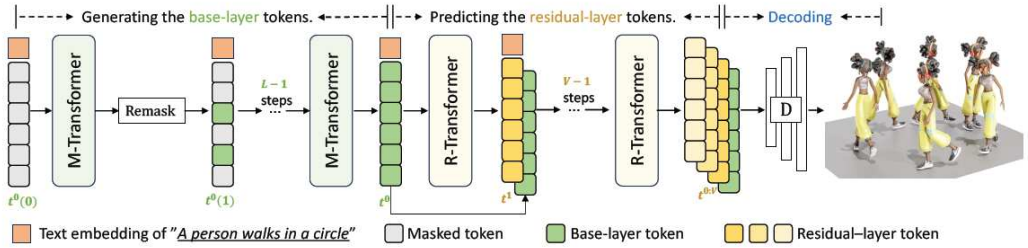


Figura 2.5: Partendo da una sequenza vuota $t_0^{(0)}$, l' M-Transformer genera la sequenza di token del layer base t^0 in L iterazioni. Successivamente, l'R-Transformer prevede progressivamente le sequenze di token degli strati residui $t^{2:V}$ in $V-1$ passaggi.[68]

L'addestramento del modello MoMask ha previsto l'utilizzo di due **dataset di movimento-linguaggio** ampiamente riconosciuti in letteratura: **HumanML3D** e **KIT-ML** [63, 60, 77].

2.4.2 ML-Agents

Il toolkit **ML-Agents** è un progetto open-source che permette di sfruttare il game engine Unity per l'addestramento di agenti intelligenti. Questo strumento rappresenta un'importante risorsa poiché mette a disposizione una vasta gamma di funzionalità e algoritmi per l'addestramento degli agenti in numerosi scenari. Oltre a ciò, offre una piattaforma unificata in cui i progressi nel campo dell'AI (intelligenza artificiale) possono essere valutati all'interno degli ambienti Unity, rendendo tali lavori accessibili alla più ampia comunità di ricerca e sviluppo nel settore dei videogiochi. Il toolkit si compone di cinque componenti principali:

- **Ambiente di apprendimento**: rappresentato dalla scena Unity, che funge da contesto in cui si svolge il ciclo di apprendimento dell'agente.
- **API Python di basso livello**: offre agli utenti la possibilità di interagire con l'ambiente di apprendimento tramite Python, permettendo il monitoraggio dell'addestramento, oltre che a facilitare l'integrazione di Unity con algoritmi di apprendimento automatico personalizzati.
- **Communicator esterno**: costituisce il collegamento tra l'ambiente di apprendimento e l'API Python di basso livello.
- **Python Trainers**: sono responsabili dell'implementazione degli algoritmi di apprendimento automatico utilizzati per addestrare gli agenti.
- **Gym Wrapper e PettingZoo Wrapper**: forniscono la possibilità di integrare gli ambienti Unity con algoritmi di apprendimento automatico già esistenti che utilizzano le API Gym o PettingZoo.

Per poter procedere con l'addestramento di un agente, è necessario definire due componenti fondamentali per l'ambiente di apprendimento:

- **Agents**: entità associata a un GameObject di Unity, che si occupa della generazione delle proprie osservazioni, dell'esecuzione delle azioni e dell'assegnazione di una ricompensa, quando opportuno. Ogni agente è associato a un comportamento *(Behavior)*, e diversi agenti all'interno dello stesso ambiente possono condividere lo stesso comportamento.

- **Behavior**: componente che riceve osservazioni e ricompense dall'agente e restituisce le azioni da eseguire. Grazie a questo elemento è possibile definire attributi specifici dell'agente, come il numero e il tipo di azioni da intraprendere.

Per il corretto funzionamento del framework, ogni ambiente di apprendimento deve sempre avere almeno un agente per ogni personaggio presente nella scena, e ogni agente deve essere associato a un comportamento Behavior. Inoltre, all'interno di un singolo ambiente possono coesistere più agenti e comportamenti. È possibile, inoltre, scambiare dati tra Unity e Python al di fuori del ciclo di apprendimento automatico tramite Side Channels. Un esempio di questo scambio può essere l'invio di parametri dell'ambiente a Python come rappresentato in Figura 2.6.

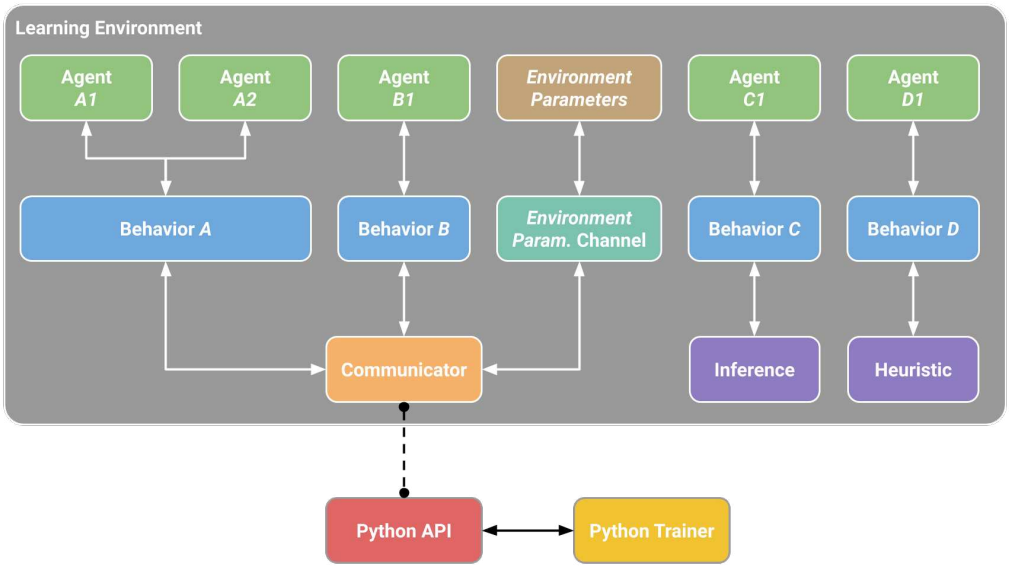


Figura 2.6: Diagramma a blocchi dell'ecosistema di ML-Agents Toolkit [78].

L'addestramento di un agente avviene attraverso il ciclo di apprendimento che rappresenta un processo iterativo in cui l'agente acquisisce progressivamente competenze per interagire con l'ambiente. Questo ciclo può essere descritto attraverso quattro fasi principali:

1. **Osservazione**: L'agente raccoglie informazioni riguardanti l'ambiente circostante tramite i sensori di cui dispone. Le osservazioni forniscono una rappresentazione dello stato dell'ambiente che sarà utilizzata nelle fasi successive.

2. **Decisione**: Sulla base delle osservazioni acquisite, l'agente è chiamato a scegliere un'azione. La decisione viene presa in accordo con la *policy* dell'agente, che può essere rappresentata da una rete neurale addestrata o da un algoritmo euristico. La *policy* processa le osservazioni e seleziona l'azione che si prevede ottimizzi al meglio la ricompensa futura.

3. **Azione**: L'agente esegue l'azione scelta dalla *policy* nell'ambiente, interagendo direttamente con esso.

4. **Ricompensa**: A seguito dell'azione eseguita, l'ambiente fornisce all'agente un segnale di ricompensa. La ricompensa è rappresentata da un valore numerico che misura quanto l'azione intrapresa ha contribuito al raggiungimento dell'obiettivo. Ricompense positive incentivano l'agente a ripetere comportamenti che conducono a esiti desiderabili, mentre ricompense negative dissuadono l'agente dal ripetere comportamenti che conducono a risultati sfavorevoli.

Questo ciclo, rappresentato sinteticamente in Figura 2.7, viene ripetuto iterativamente durante l'intero processo di addestramento. Ad ogni iterazione, l'agente sfrutta il segnale di ricompensa per aggiornare e migliorare la propria *policy*, con l'obiettivo di aumentare le proprie prestazioni. Il fine ultimo dell'agente è apprendere una *policy* in grado di massimizzare la ricompensa cumulativa a lungo termine, garantendo così un comportamento ottimale nell'interazione con l'ambiente.

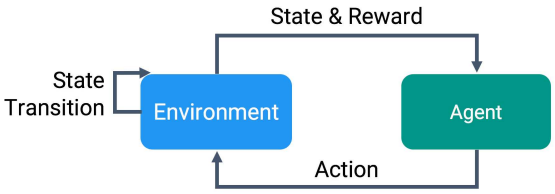


Figura 2.7: Diagramma a blocchi del ciclo di apprendimento di un agente [79].

Capitolo 3

Progettazione e sviluppo

La fase di progettazione, con il conseguente sviluppo del progetto, è stata guidata dagli obiettivi di ricerca del lavoro, con l'intento di realizzare un sistema che permettesse a un **manichino umanoide** di **apprendere ed eseguire movimenti naturali all'interno di una scena virtuale**. Questi movimenti, controllati da un modello intelligenza artificiale, devono essere eseguiti con un elevato grado di realismo, rispecchiando le caratteristiche biomeccaniche e i comportamenti tipici del corpo umano.

Per il raggiungimento di questi obiettivi, è stata adottata una metodologia iterativa, basata su cicli di progettazione, implementazione e test. Ogni fase di sviluppo è stata seguita da verifiche sperimentali per migliorare progressivamente le prestazioni del sistema. In particolare, il processo ha seguito i seguenti step principali:

- Progettazione di un'architettura modulare e scalabile, in grado di supportare diversi modelli biomeccanici e configurazioni.
- Implementazione delle principali componenti del sistema, tra cui:
  - Importazione dei modelli **SMPL** per la rappresentazione di percentili umanoidi differenti.
  - Definizione di un **ragdoll biomeccanico** per l'interazione del manichino con l'ambiente e la fisica.
  - **Caricamento dei dati** per l'imitazione dei movimenti per l'addestramento del modello.
  - Implementazione della **logica dell'agente** per l'apprendimento e l'esecuzione dei movimenti.
- Testing e validazione delle soluzioni sviluppate, con valutazioni sulla stabilità del manichino, la qualità delle animazioni e l'efficacia dell'apprendimento.

Per comprendere meglio il flusso operativo del sistema sviluppato, la Figura 3.1 mostra l'architettura generale, evidenziando l'interazione tra i diversi componenti coinvolti nel processo di generazione e addestramento del manichino umanoide. Lo schema illustra il percorso dei dati, partendo dall'inserimento di un prompt testuale in Unity fino al monitoraggio dell'addestramento tramite TensorBoard, delineando il ruolo di ciascun modulo e le connessioni tra di essi.

Come descritto nei capitoli precedenti, l'addestramento è stato eseguito utilizzando il framework **ML-Agents**, integrato direttamente nell'ambiente di simulazione di Unity. Il modello è stato ottimizzato attraverso **Proximal Policy Optimization (PPO)**, un algoritmo di policy gradient ampiamente utilizzato per addestrare agenti nei problemi di reinforcement learning [72].

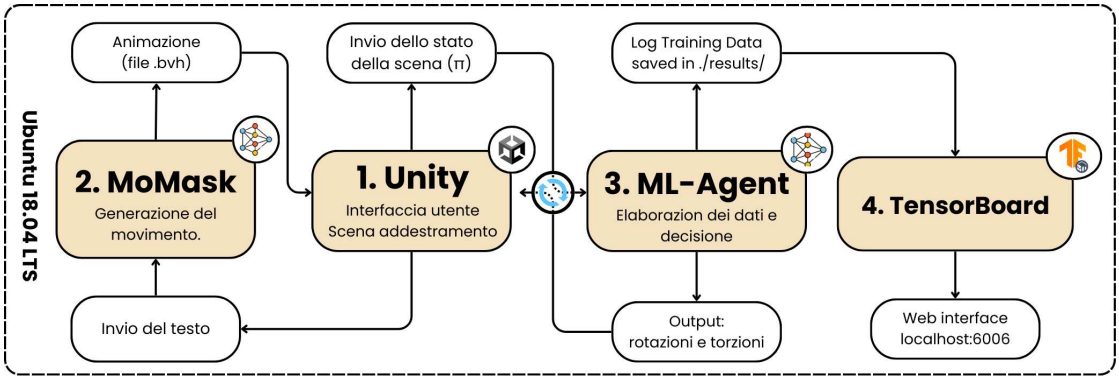


Figura 3.1: Il processo inizia con l'inserimento di un prompt testuale in Unity, che viene inviato a MoMask per generare un'animazione in formato .bvh. Il file animato viene caricato in Unity, dove viene utilizzato nel loop di addestramento con ML-Agents. Durante l'addestramento, Unity invia lo stato corrente della scena all'agente AI, che elabora i dati e genera come output nuove rotazioni e torsioni per il manichino. I dati relativi all'andamento dell'addestramento vengono salvati in TensorBoard, accessibile tramite interfaccia web sulla porta 6006, per il monitoraggio delle metriche di training.

In seguito, verranno presentate le scelte progettuali adottate e i dettagli delle fasi di implementazione e testing.

3.1 Preprocessing dei dati

L'addestramento di un agente umanoide basato sull'**imitazione di movimenti** richiede una fase preliminare di **preprocessing dei dati**, necessaria per trasformare le animazioni target in un formato compatibile con il sistema di apprendimento dell'agente.

Nel contesto del progetto, si è scelto di utilizzare **animazioni generate dal modello MoMask**, così da poter disporre di movimenti specifici e personalizzati in base al contesto di lavoro.

Questa sezione descrive nel dettaglio le fasi chiave del processo di preprocessing, partendo dalla generazione dell'animazione a partire da un prompt testuale fino alla conversione e strutturazione dei dati per il training dell'agente. La Figura 3.2 schematizza il flusso operativo di questa fase, illustrando il percorso dei dati dalla loro origine fino alla loro integrazione nel sistema di apprendimento.

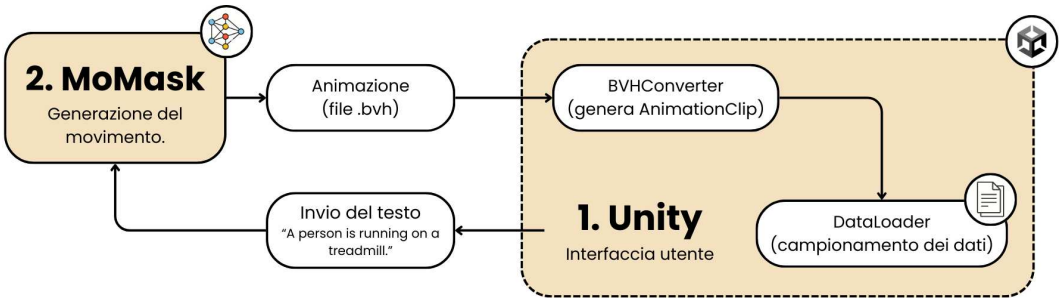


Figura 3.2: Flusso di preprocessing dei dati di animazione per l'addestramento dell'agente. Il processo inizia all'interno di Unity, dove un'interfaccia utente consente di inserire un prompt testuale che descrive il movimento desiderato. Questo input viene inviato al modello di intelligenza artificiale MoMask, il quale genera un'animazione corrispondente salvata in formato BVH. Successivamente, il file BVH viene processato dal BVHConverter, che lo trasforma in un AnimationClip utilizzabile in Unity. Infine, il DataLoader esegue il campionamento dell'animazione convertita, strutturando i dati in un formato adatto all'addestramento dell'agente.

3.1.1 Generazione dell'animazione a partire da un prompt testuale in Unity

Il processo di addestramento dell'agente umanoide inizia dall'**inserimento di un prompt testuale**. Questo dato viene utilizzato come input per il modello di intelligenza artificiale Momask, il quale **genera un'animazione target** che dovrà essere imitata. Essa costituisce il riferimento per la fase di training, fornendo una sequenza di movimenti realistici che il modello dovrà apprendere e riprodurre. Per la sua implementazione, all'interno del sistema Unity, è stato sviluppato un componente MonoBehaviour, che consente all'utente di definire un movimento desiderato attraverso linguaggio naturale. Ad esempio, un tipico **comando** per la generazione dell'animazione desiderata potrebbe essere:

A person grasping an object above a table at pelvis height.

Oltre all'inserimento del prompt testuale, l'utente ha la possibilità di selezionare il tipo di dispositivo hardware su cui eseguire il processo di generazione scegliendo fra GPU o CPU. Quando l'utente avvia la generazione, Unity esegue lo script Python `gen_t2m.py`, utilizzando un ambiente Conda. Il comando completo è costruito dinamicamente e include i seguenti parametri:

- **gpu_id**: Specifica quale GPU utilizzare per l'inferenza.
- **ext**: Identificativo dell'esperimento, utile per organizzare i risultati.
- **text_prompt**: Il testo che descrive il movimento da generare.

L'invocazione del processo avviene tramite la classe `ProcessStartInfo` definita in C#, che permette di eseguire lo script Python direttamente da Unity. Se il processo viene eseguito con successo, MoMask genera un file .bvh contenente l'animazione desiderata. La generazione dell'animazione target rappresenta il primo passo nel processo di addestramento dell'agente. Tuttavia, affinché il file generato possa essere utilizzato efficacemente, è necessario effettuare il **parsing dei dati** ottenuti. Questo processo consente di interpretare e strutturare i dati di movimento in un formato compatibile con Unity, garantendo che le informazioni di rotazione estratte possano essere applicate correttamente al manichino umanoide.

3.1.2 Parsing e conversione dei file BVH in AnimationClip

Il processo di parsing dei file BioVision Hierarchy nel progetto è organizzato in più fasi per leggere, interpretare e convertire i dati di animazione contenuti nei file in un formato compatibile con Unity. L'integrazione della libreria open source disponibile su GitHub [80] ha permesso di estrarre le informazioni contenute nel file **BVH**, convertirle in curve di animazione e salvare il risultato come un **AnimationClip**, asset compatibile con Unity.

L'operazione inizia con la lettura del file BVH, il quale contiene la gerarchia delle ossa del manichino e i dati di movimento registrati come trasformazioni nel tempo. Il parser è in grado di operare in due modalità: rispettando il frame time originale definito nel BVH o utilizzando un frame rate predefinito dall'utente. Una volta inizializzato il parser e caricate le informazioni di animazione, il sistema genera un oggetto di tipo AnimationClip, che rappresenterà la sequenza animata in Unity. Questo oggetto viene configurato con il nome del file BVH.

Successivamente, il codice si sviluppa gestendo la generazione delle curve di animazione, un'operazione che avviene ricorsivamente su tutti i nodi della gerarchia scheletrica del file BVH. Per ogni osso, il metodo `GetCurves` analizza i canali di trasformazione disponibili e stabilisce quali informazioni di rotazione sono presenti. Le rotazioni vengono poi convertite in frame chiave (Keyframe) per ogni istante di tempo, in modo da poter ricostruire fedelmente il movimento nel tempo.

Poiché i file BVH utilizzano il sistema di rotazione basato su angoli di Eulero con ordinamento ZXY, le rotazioni vengono convertite in quaternioni, che sono il formato standard per Unity. Una volta generate le curve di animazione, la posizione del bacino e le rotazioni locali di ciascun giunto rispetto al proprio nodo genitore vengono mappate ai corrispondenti parametri dell'oggetto Transform del modello in Unity. Infine, una volta completata la conversione dei dati di animazione, l'AnimationClip risultante viene salvato all'interno della directory specificata.

In aggiunta, è stata sviluppata un'interfaccia utente dedicata nell'editor di Unity, gestita dalla classe `BVHConverterEditor`. Questa interfaccia consente agli utenti di selezionare una directory contenente i file BVH e di configurare parametri come il rispetto del frame time originale, offrendo un controllo più diretto e intuitivo sulla conversione dei file di animazione. In Figura 3.3 è possibile avere una rappresentazione visiva di come è organizzata l'interfaccia.

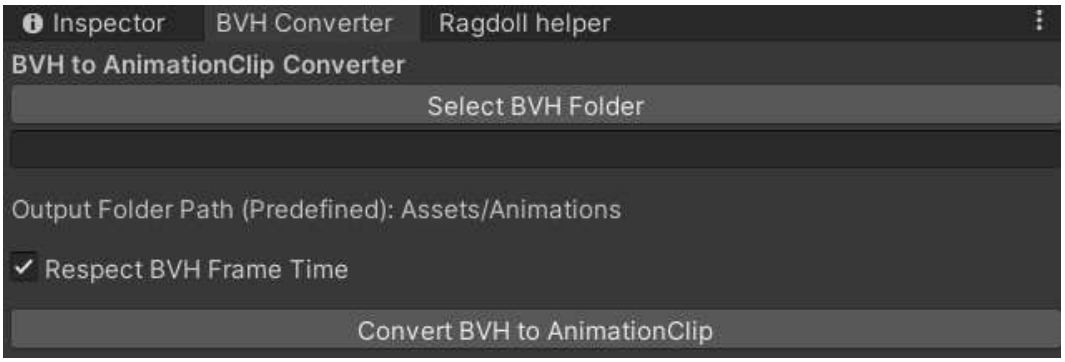


Figura 3.3: Editor della libreria bvhConverter, che si occupa di fare il parsing da un insieme di file BVH presenti all'interno di una cartella selezionata dall'utente, ad un insieme di AnimationClip per Unity.

Una volta completata la conversione dei file BVH in *AnimationClip*, è necessario estrarre i dati rilevanti per l'addestramento dell'agente.

3.1.3 Caricamento dei dati

Effettuata la conversione del file BVH in AnimationClip, è necessario estrarre e strutturare le informazioni in un formato ottimizzato per il training. Questo passaggio permette di trasformare le animazioni in dati accessibili e interpretabili dall'agente.

La classe progettata per gestire il caricamento e l'organizzazione dei dati di animazione è denominata `DataLoader`. Essa implementa un sistema basato su dizionari, consentendo non solo la gestione efficiente dei dati, ma anche il caricamento simultaneo di più animazioni. Questa funzionalità apre la possibilità di addestrare

l'agente su un insieme più ampio di movimenti, rappresentando un'estensione naturale per sviluppi futuri.

Ogni animazione è identificata da una chiave corrispondente al nome del file di origine, mentre i dati associati vengono organizzati in una struttura nidificata. In particolare, i nomi dei giunti sono mappati a liste di tuple, dove ogni tupla contiene la posizione (*Vector3*) e la rotazione (*Quaternion*) del giunto in un determinato frame.

Il caricamento delle animazioni avviene a partire da file *AnimationClip*. Per ciascun file, un estrattore specializzato, `AnimationClipDataExtractor`, si occupa di leggere e interpretare le curve di animazione, traducendole in una struttura dati adatta all'addestramento. Uno degli aspetti fondamentali della classe `DataLoader` è il processo di campionamento delle curve di animazione. Poiché le curve rappresentano variazioni continue nel tempo, è necessario campionarle a intervalli regolari in base al frame rate desiderato, in modo da ottenere una sequenza discreta di frame, come illustrato in Figura 3.4.

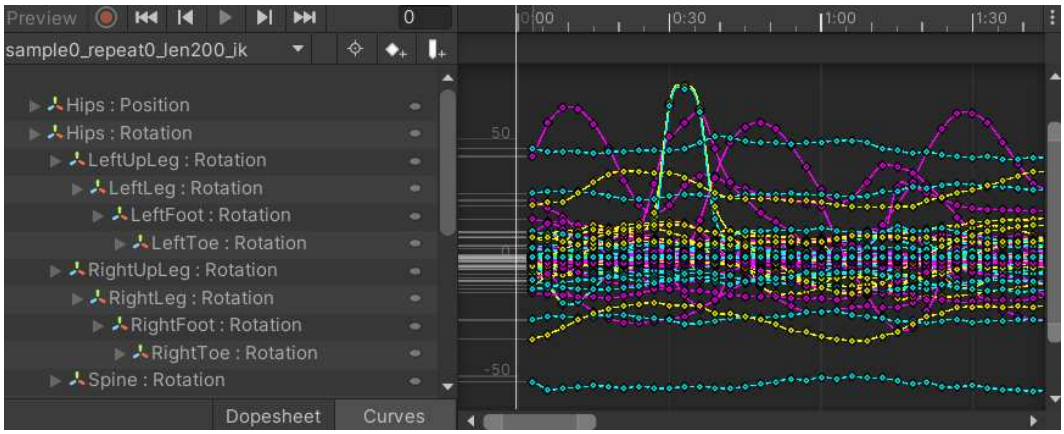


Figura 3.4: Il processo di campionamento implementato nel `DataLoader` consente di trasformare i dati di animazione da una rappresentazione continua nel tempo a una sequenza discreta di frame.

3.2 Definizione del manichino umanoide

Per effettuare simulazioni realistiche all'interno di un ambiente virtuale in Unity, sfruttando le potenzialità del motore fisico integrato nel game engine, è necessario definire un ragdoll capace di simulare il comportamento del corpo umano quando soggetto a forze fisiche, come la gravità. Questo elemento è stato integrato con i modelli parametrici Skinned Multi-Person Linear Model, grazie all'utilizzo del

framework **Ragdoll Helper** [81], che ha consentito la definizione automatica del ragdoll. L'architettura è illustrata in Figura 3.5.

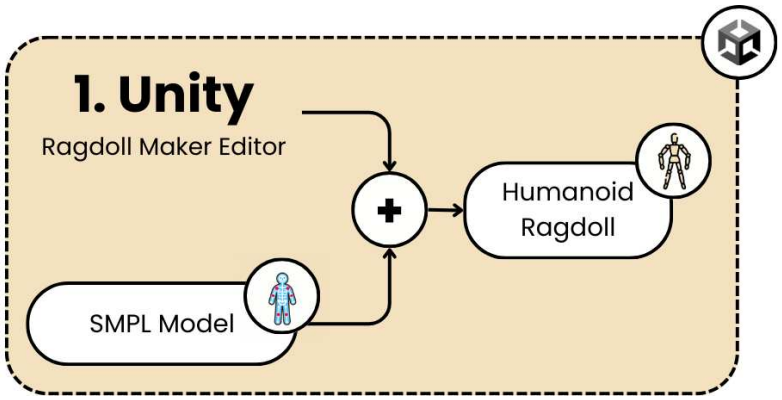


Figura 3.5: Architettura del sistema per la generazione del ragdoll umanoide in Unity. Il Ragdoll Maker Editor consente di combinare un modello parametrico SMPL con una struttura fisica articolata, generando automaticamente un Humanoid Ragdoll compatibile con il motore fisico di Unity.

Di seguito verranno descritte le operazioni di implementazione relative all'utilizzo dei modelli SMPL nell'ambiente Unity e alla definizione del ragdoll biomeccanico, essenziale per consentire l'interazione dell'umanoide con la fisica del sistema.

3.2.1 Importazione dei modelli SMPL in Unity

Nel contesto di questo progetto, la scelta del modello SMPL è stata guidata dalla sua capacità di parametrizzare il corpo umano attraverso i **parametri di shape**, che descrivono diverse variabilità anatomiche, permettendo di creare manichini personalizzabili, capaci di rappresentare differenti percentili antropometrici. Un esempio sull'impatto che i parametri di shape hanno sul manichino è mostrato in Figura 3.7. Inoltre, i parametri di shape possono essere salvati in file *.json*, consentendo di mantenere configurazioni specifiche e riutilizzabili in futuro. Un ulteriore punto di forza del modello SMPL è l'uso del sistema di Linear Blend Skinning per deformare la mesh in base alla posizione delle articolazioni, aumentando la fedeltà visiva del movimento.

Per importare i modelli SMPL in una scena di Unity, è necessario scaricare i file e gli script dal sito ufficiale [82]. Una volta ottenuti, i file devono essere collocati all'interno della directory `"Assets"` del progetto Unity, preferibilmente in una cartella dedicata.

I modelli FBX di SMPL (maschile e femminile) verranno caricati automaticamente e saranno visibili nella finestra `"Project"`. Successivamente, è necessario

configurare correttamente i modelli accedendo alle impostazioni di importazione del file FBX, come illustrato in Figura 3.6. Un aspetto fondamentale è la selezione di `Humanoid` come tipo di animazione, garantendo così la compatibilità con il sistema Mecanim di Unity.

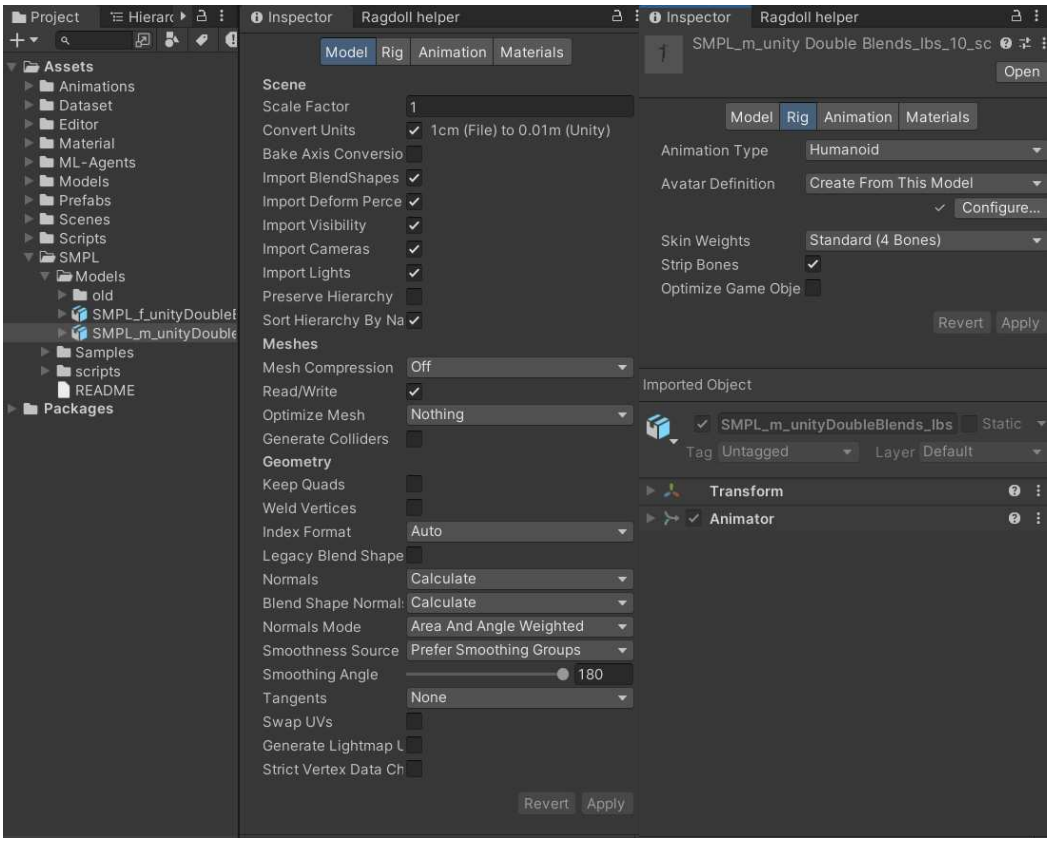


Figura 3.6: Configurazione dei modelli SMPL nelle impostazioni di importazione di Unity.

Completata la configurazione, i modelli possono essere trascinati dalla finestra `"Project"` alla finestra `"Hierarchy"` per essere aggiunti alla scena. Da qui, è possibile personalizzarli ulteriormente aggiungendo i componenti forniti da Unity.

All'interno della cartella `Assets/SMPL/scripts/MPI` sono presenti alcuni script che facilitano la gestione del modello SMPL:

- `SMPLOptimalPoseBlends`: Ottimizza i blendshape delle pose del modello SMPL per migliorare le prestazioni, in particolare in contesti come la realtà virtuale. Riduce il numero di blendshape considerati, selezionando solo quelli più rilevanti per un effetto visivo ottimale, contribuendo così a mantenere un framerate elevato.

- `SMPLBlendshapes`: Gestisce i blendshape di forma e posa del modello SMPL. Consente di aggiornare la mesh per riflettere specifiche caratteristiche corporee, regolando proporzioni e dettagli anatomici. Inoltre, applica effetti realistici in base alla posa, come pieghe articolari o rigonfiamenti muscolari, e supporta l'integrazione di file JSON per la personalizzazione delle configurazioni.
- `SMPLJointCalculator`: Calcola la posizione delle articolazioni del modello SMPL in base ai parametri di forma. Utilizza regressori di giunti specificati nei file JSON per determinare le posizioni aggiornate delle articolazioni sulla mesh.
- `SMPLModifyBones`: Gestisce la modifica delle ossa del modello SMPL in base alle variazioni di forma e posa. Aggiorna dinamicamente la posizione e l'orientamento delle articolazioni per garantire una rappresentazione fedele delle trasformazioni applicate. Inoltre, assicura che il modello rimanga ancorato al terreno e che la configurazione iniziale sia mantenuta correttamente.

Figura 3.7: Visualizzazione dello stesso modello SMPL con parametri di forma differenti, gestiti dal componente `SMPLBlendshapes`.

3.2.2 Definizione del ragdoll

Un elemento importante per ottenere movimenti naturali e realistici, nonché per consentire l'interazione del modello con la scena, è il **ragdoll biomeccanico.** Si tratta di una struttura costituita da articolazioni e componenti fisici, progettata per simulare il comportamento del corpo umano in risposta a forze esterne, come la gravità e le interazioni con l'ambiente. In questo progetto, il ragdoll è stato integrato con il modello parametrico SMPL, sfruttandone la struttura scheletrica per realizzare un manichino altamente aderente alle esigenze applicative.

Per semplificare la creazione e configurazione del ragdoll in Unity, è stata utilizzata la libreria **Ragdoll Helper** [81], successivamente personalizzata per adattarla alle esigenze. La libreria consente l'assegnazione automatica dei componenti necessari alla struttura e al comportamento del ragdoll. Affinché il sistema funzioni correttamente, l'oggetto a cui viene applicato deve avere l'**Animation Type** impostato su **Humanoid** per garantirne la corretta compatibilità. Ogni giunto di interesse viene dotato di un **Collider**, un **RigidBody** e un **Character Joint**, elementi essenziali per la simulazione fisica del ragdoll.

Nel progetto sono stati configurati tre giunti principali per il tronco (Hips, Spine2, Head), oltre ai giunti simmetrici degli arti inferiori (Hip, Knee, Ankle) e superiori (Shoulder, Elbow, Wrist), come è possibile vedere in Figura 3.8.

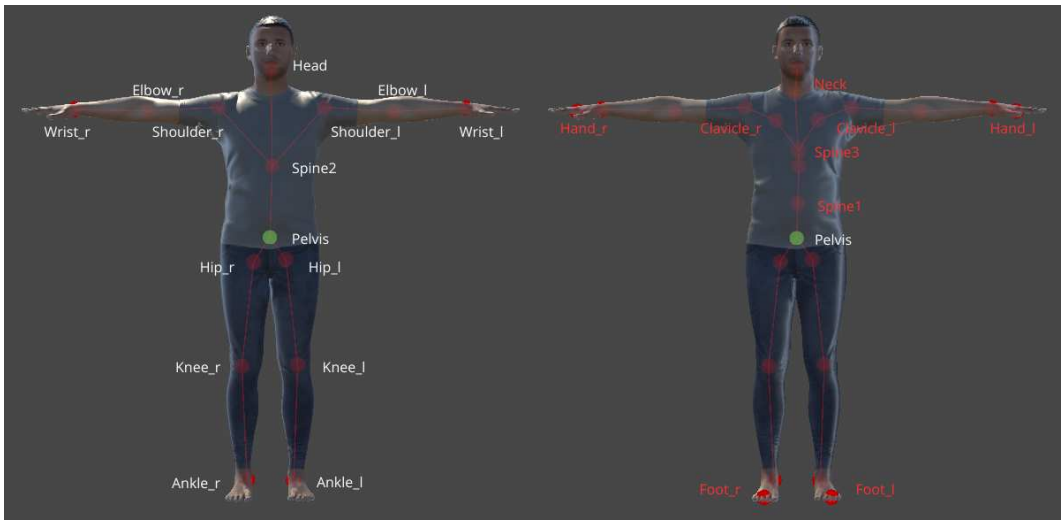


Figura 3.8: Confronto tra la gerarchia dei giunti del ragdoll (sinistra) e la gerarchia estratta dal file BVH (destra). I giunti del ragdoll sono stati configurati per l'interazione fisica nell'ambiente Unity, mentre la gerarchia BVH conserva la struttura originale dell'animazione, includendo giunti aggiuntivi come il collo, le clavicole e le mani.

La scelta di limitare il numero di giunti rispetto a quelli disponibili è stata guidata da considerazioni di efficienza computazionale. In particolare, nel tronco sono stati esclusi quattro giunti (due per la spina dorsale e due per le clavicole). Sebbene la loro inclusione sarebbe stata importante per una simulazione ergonomica più accurata, il loro costo computazionale (24 gradi di libertà moltiplicati per il numero di agenti in addestramento) ha reso preferibile una soluzione temporanea che ne prevede l'esclusione, bilanciando così fedeltà della simulazione e prestazioni.

La libreria *Ragdoll Helper* offre un certo grado di automazione, rendendo più efficiente il flusso di lavoro. Una volta creato il ragdoll, gli strumenti inclusi permettono di modificare i *Collider* in modo simmetrico tra le parti destra e sinistra del corpo, semplificando operazioni di rotazione, spostamento e scalatura. Inoltre, include un'interfaccia utente integrata in Unity che fornisce un pannello intuitivo per la gestione e la personalizzazione del ragdoll. I parametri configurabili e il risultato della generazione sono illustrati in Figura 3.9.

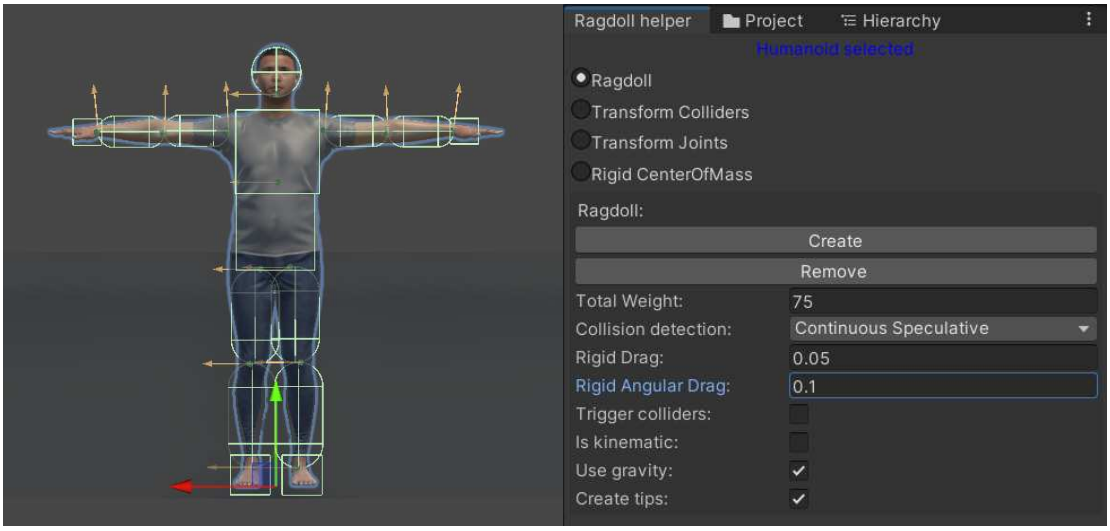


Figura 3.9: Configurazione del Ragdoll Helper.

Per migliorare il controllo sui limiti articolari e sulle forze applicate ai giunti, il componente Character Joint è stato sostituito con un **Configurable Joint**, che offre una maggiore flessibilità nella regolazione dei parametri biomeccanici. Anche in questa fase, si è mantenuto un alto grado di automazione, evitando che l'operazione debba essere eseguita manualmente all'interno dell'editor.

La gestione dei limiti angolari per ogni giunto è elaborata attraverso il metodo `ConfigureJointLimits()`. Questo prende in ingresso un giunto e i relativi valori numerici per definire i limiti angolari e di torsione. I movimenti lineari del giunto vengono completamente bloccati, mentre per i movimenti angolari vengono impostati limiti precisi su ciascun asse per garantire che il giunto operi entro vincoli

realistici. Inoltre, il metodo configura la rigidità e lo smorzamento del movimento angolare, contribuendo a rendere il comportamento del giunto più naturale durante le interazioni fisiche.

I valori utilizzati per descrivere il comportamento di torsione e i limiti angolari di ciascun giunto sono riportati in Tabella 3.1.

Nome del Giunto	**lowTwist (°)**	**highTwist (°)**	**swing1 (°)**	**swing2 (°)**
Head	-45	45	60	60
Spine2	-30	30	20	20
Pelvis	-15	15	25	25
Shoulders	-60	60	120	120
Elbows	0	145	0	0
Wrists	-20	20	45	45
Hips	-30	30	60	60
Knees	0	130	0	0
Ankles	-20	20	30	30

Tabella 3.1: Valori inseriti per le giunture del corpo umano.

Al termine del processo di configurazione, ogni giunto risulta dotato dei seguenti componenti, che possono essere ulteriormente ottimizzati in fase di simulazione:

- **Collider**: definisce la geometria fisica delle parti del corpo.
- **Rigid Body**: gestisce l'applicazione delle leggi della fisica.
- **Configurable Joint**: consente la simulazione dei movimenti articolari.

3.3 Definizione dell'agente

L'agente è stato progettato e implementato per apprendere movimenti complessi adottando un approccio che integra imitazione del movimento e controllo orientato al compito, all'interno dell'ambiente di simulazione di Unity, sfruttando il framework ML-Agents. La logica di apprendimento dell'agente è stata sviluppata prendendo come riferimento il lavoro svolto nel progetto DeepMimic [72].

Il comportamento dell'agente è definito all'interno della classe `HumanoidAgent`, che si basa su una gerarchia di giunti articolari rappresentata da `BodyPart`. La gestione della logica dei movimenti articolari del manichino è affidata alla classe `JointDriveController`, responsabile del controllo delle dinamiche dei giunti durante l'esecuzione dei movimenti. Mentre la gestione dei dati relativi all'animazione e la logica relativa al completamento del task sono lasciate rispettivamente alle classi `HumanoidCloningManager` e `HumanoidRLManager`.

Come descritto nei capitoli precedenti, l'agente apprenderà attraverso **cicli di apprendimento** strutturati in quattro fasi principali: **osservazione**, **decisione**, **azione** e **ricompensa** come mostrato in Figura 3.10. Nei capitoli successivi verranno analizzati in dettaglio i principali componenti che costituiscono l'agente e le specifiche modalità con cui è stato implementato il ciclo di apprendimento.

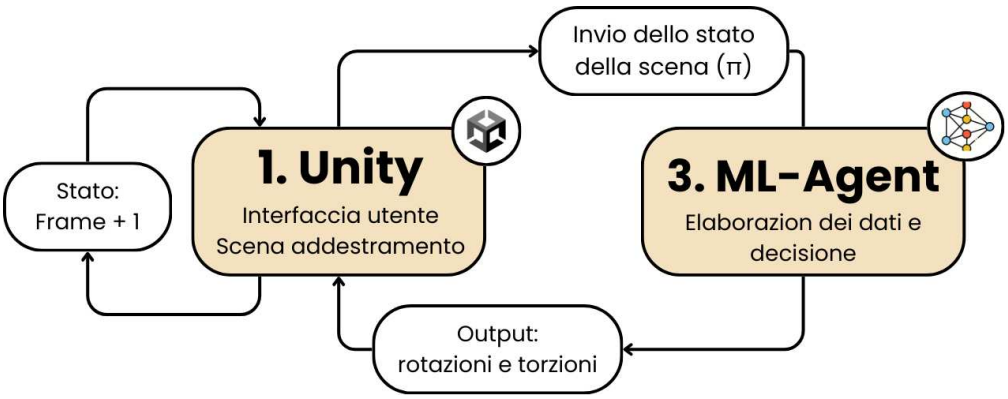


Figura 3.10: Schema del ciclo di apprendimeto di un agente che imita un'animazione di movimento utilizzando Unity e ML-Agents. Il processo inizia con Unity, che gestisce l'interfaccia utente e la simulazione della scena di addestramento. Lo stato della scena viene inviato all'agente di apprendimento (**osservazione**) (ML-Agent), che elabora i dati (**decisione**) e prende decisioni basate sul modello (**decisione**). L'output generato, costituito da rotazioni e torsioni, viene restituito a Unity per aggiornare lo stato del personaggio (**ricompensa**). Il ciclo si ripete iterativamente, avanzando di un frame alla volta, fino al completamento dell'addestramento.

3.3.1 HumanoidAgent

La classe `HumanoidAgent` estende la classe **Agent** del framework **ML-Agents** e costituisce il nucleo della logica di addestramento e comportamento del manichino umanoide. Questa classe gestisce l'interazione dell'agente con l'ambiente virtuale, definendo le modalità con cui apprende e riproduce i movimenti. Il suo comportamento è organizzato attraverso una sequenza di metodi che gestiscono l'inizializzazione, la raccolta di dati, l'applicazione delle azioni e la gestione delle ricompense. Di seguito verrà data una panoramica dei metodi principali implementati all'interno della classe.

Awake

La presente classe implementa il metodo di **inizializzazione delle informazioni necessarie per l'addestramento** dell'agente. In più gestisce la configurazione dei componenti essenziali e la regolazione dei parametri temporali per l'animazione e la simulazione fisica. In fase di avvio, il sistema verifica la presenza dei componenti `HumanoidCloningManager` e `HumanoidRLManager`.

Dopo aver accertato la disponibilità di entrambi, il metodo `LoadAnimation()` provvede al **caricamento del set di animazioni**. Successivamente, la struttura scheletrica dell'animazione viene adattata alla gerarchia della scena e il frame rate dell'animazione viene quindi acquisito e utilizzato per il **calcolo della velocità angolare dei giunti**.

Al completamento dell'inizializzazione, i parametri necessari alla simulazione del movimento sono correttamente caricati e pronti all'uso. La Figura 3.11 mostra le diverse fasi di elaborazione dell'animazione applicata al manichino virtuale.

Initialize

Ha il compito di configurare i componenti chiave del modello. Tra questi, sono presenti l' `OrientationCubeController`, un GameObject che fornisce un riferimento stabile per la raccolta delle osservazioni nella scena durante tutto l'addestramento e la classe `JointDriveController`, che gestisce i giunti articolari del manichino. Durante questa fase, vengono inoltre impostate le posizioni iniziali del bacino e le forze di stabilizzazione, fondamentali per mantenere il manichino stabile durante l'apprendimento.

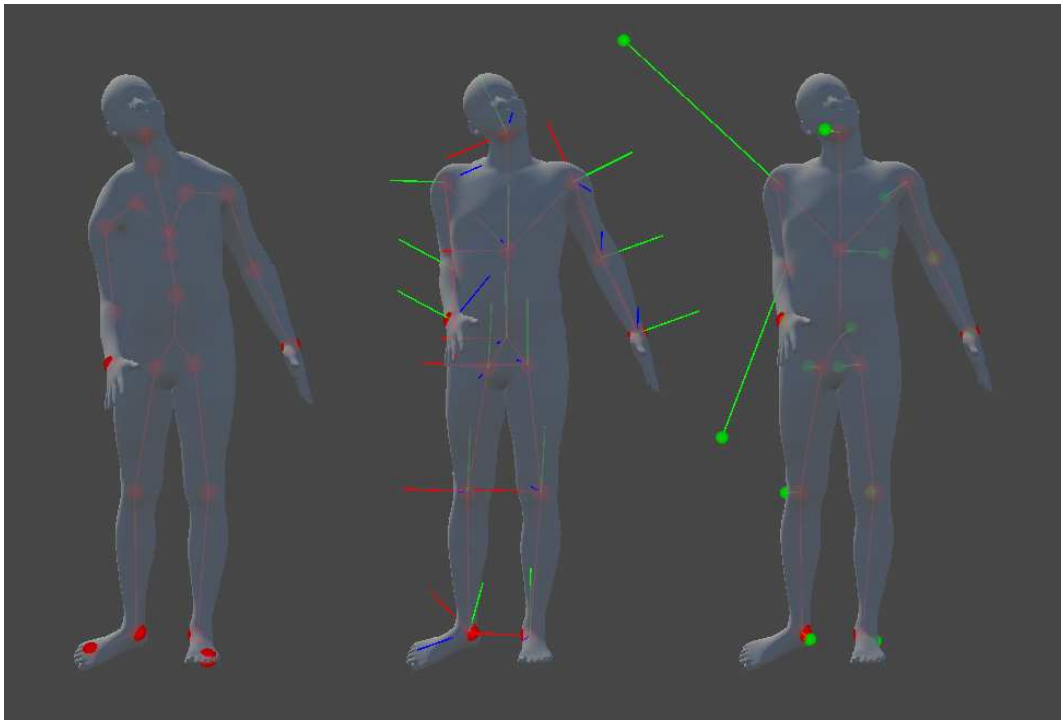


Figura 3.11: Da destra a sinistra si osservano tre fasi del processo di elaborazione dell'animazione: la prima rappresenta la struttura scheletrica estratta dal file BVH grazie al `DataLoader` con la rispettiva animazione applicata, la seconda mostra l'animazione riadattata alla struttura del manichino presente nella scena, e la terza include la visualizzazione delle velocità angolari calcolate per ciascun giunto.

OnEpisodeBegin

Ogni episodio di addestramento inizia con il metodo `OnEpisodeBegin`, che ripristina lo stato iniziale dell'agente resettando la posizione e la rotazione di ogni giunto. Come dichiarato nel lavoro DeepMimic, anche in questo progetto è stata implementata la **Reference State Initialization (RSI)**, una strategia che prevede l'inizializzazione dello stato dell'agente attraverso il campionamento casuale di un frame all'interno dell'animazione di riferimento. Questo approccio consente di variare le condizioni di partenza dell'agente, migliorando la stabilità dell'apprendimento e favorendo una maggiore generalizzazione dei movimenti appresi [72].

CollectObservations

Il metodo `CollectObservation`, ereditato dalla classe Agent, rappresenta la prima fase del ciclo di apprendimento dell'agente. Durante questa fase, il metodo raccoglie dati dall'ambiente, fornendo all'agente informazioni essenziali per prendere decisioni. Nel contesto del progetto, le osservazioni includono:

- **Variabili globali**:
  - Una variabile di processo che indica il punto in cui ci troviamo nell'animazione.
  - Posizione del target.
  - Una Variabile binaria, che indica se il task è stato compiuto precedentemente.
  - Distanza della posizione delle mani e il target.
  - Posizione del centro di massa.
  - Orientamento del bacino e della testa
- **Osservazioni specifiche per ogni giunto**:
  - Interazione del giunto con l'ambiente.
  - Posizione del giunto rispetto il bacino.
  - Rotazione locale del giunto.
  - Velocità angolari e lineari del giunto.
  - Forza applicata al giunto.

Tutte le osservazioni sono raccolte rispetto al sistema di riferimento del GameObject `OrientationCubeController`. Questo approccio garantisce stabilità nei dati acquisiti e consente l'addestramento di più agenti all'interno dello stesso ambiente, grazie all'uso di un sistema di riferimento locale.

OnActionReceived

Il metodo `OnActionReceived`, anch'esso ereditato dalla classe `Agent`, rappresenta la fase di azione nel ciclo di apprendimento dell'agente. In questa fase, le azioni generate dalla rete neurale vengono elaborate e tradotte in comandi motori per il manichino. I valori ricevuti in input vengono interpretati e convertiti in rotazioni target e forze applicate ai giunti, consentendo il movimento dell'agente.

Per ciascun giunto, le rotazioni vengono applicate su tutti i gradi di libertà disponibili, mentre le forze vengono regolate per favorire la dinamica del movimento. Il metodo riceve come input una struttura che incapsula le azioni prodotte dal modello.

FixedUpdate

Come illustrato nel Codice 1, il metodo `FixedUpdate` valuta lo stato dell'agente durante la simulazione fisica, assegnando ricompense o penalità in base al suo comportamento. In particolare, questa fase consente di monitorare le prestazioni del modello nel raggiungimento del task assegnato.

Una delle prime verifiche effettuate riguarda il contatto tra parti specifiche dell'agente, come testa e busto, e il pavimento della scena. Se questa condizione si verifica, ovvero se la variabile indicatrice del contatto C_t assume valore 1, l'agente viene penalizzato con un punteggio di:

$$r_t^{task} = -1, \quad \text{se } C_t = 1$$

poiché tale evento è considerato critico, come indicato anche nel lavoro **DeepMimic**. In questa circostanza, l'episodio viene immediatamente terminato e ne viene avviato uno nuovo.

Se invece l'agente non entra in contatto con il pavimento ($C_t = 0$), il metodo procede calcolando la posizione del suo centro di massa nella scena. Successivamente, viene verificato se il target è stato raggiunto. Nel caso in cui l'agente tocchi il target, ovvero se la variabile T_t assume valore 1, gli viene assegnato il punteggio massimo di:

$$r_t^{task} = 1, \quad \text{se } T_t = 1$$

Se il target non è stato raggiunto ($T_t = 0$), la ricompensa viene calcolata in base alla distanza d_t tra l'end effector, ovvero la mano, e il target, con l'obiettivo di incentivare il progressivo avvicinamento all'obiettivo:

$$r_t^{task} = \exp(-4d_t), \quad \text{se } T_t = 0$$

I valori di reward calcolati in questa fase non vengono immediatamente assegnati al modello, ma vengono memorizzati per essere utilizzati successivamente nel calcolo complessivo della ricompensa.

Update

Il metodo `Update` Codice 1 viene invocato a ogni frame, monitorando lo **stato attuale** dell'agente, confrontandolo con quello **desiderato** e calcolando la ricompensa associata, al fine di guidarne l'apprendimento.

Ad ogni chiamata, viene eseguito un controllo iniziale sullo stato temporale sia dell'agente che dell'animazione di riferimento, al fine di verificare che l'agente non abbia superato la durata prevista dell'animazione. Qualora questa condizione venga violata, si considera che il compito non sia stato completato entro il tempo stabilito e l'episodio viene interrotto anticipatamente.

Invece se la condizione viene rispettata, segue l'estrazione dello stato attuale dell'agente, comprendente le **rotazioni articolari**, le **velocità angolari** e la **posizione del centro di massa**. In parallelo, viene acquisito lo stato target, prelevato dal dataset di riferimento, che include gli stessi parametri.

Confrontando questi due insiemi di dati, il metodo calcola tre ricompense parziali definite come la deviazione fra i dati raccolti in scena e quelli attesi:

- **Punteggio di posa**: calcolato come una misura angolare in radianti della differenza tra i quaternioni di orientamento dei giunti del personaggio simulato $\mathbf{q}_t^{\text{current}} = \{q_0, q_1, \ldots, q_{n-1}\}$ e quelli dell'animazione di riferimento $\mathbf{q}_t^{\text{target}} = \{\hat{q}_0, \hat{q}_1, \ldots, \hat{q}_{n-1}\}$:

$$r_t^{\text{pose}} = \exp\left[-2 \cdot \sum_{i=0}^{n-1} \|\hat{q}_i \ominus q_i\|^2\right]$$

- **Punteggio delle velocità angolari**: calcolato dalla differenza tra le velocità angolari locali dei giunti del personaggio simulato $\omega_t^{\text{current}} = \{\omega_0, \omega_1, \ldots, \omega_{n-1}\}$ e quello di riferimento $\omega_t^{\text{target}} = \{\hat{\omega}_0, \hat{\omega}_1, \ldots, \hat{\omega}_{n-1}\}$.

$$r_t^{\text{velocity}} = \exp\left[-0.1 \sum_{i=1}^{n-1} \|\hat{\omega}_i - \omega_i\|^2\right]$$

- **Punteggio del centro di massa**: quantifica la deviazione del centro di massa simulato $\hat{\vec{r}}_t$ rispetto alla posizione attesa $\vec{r}_t$.

$$r_t^{com} = \exp\left[-10 \left\|\hat{\vec{r}}_t - \vec{r}_t\right\|^2\right]$$

Queste componenti vengono combinate per generare il punteggio **di imitazione**, che rappresenta la qualità dell'imitazione in quel frame:

$$r_t^{\text{imitation}} = \lambda_p \, r_t^{\text{pose}} + \lambda_v \, r_t^{\text{velocity}} + \lambda_c \, r_t^{com}$$

A questa viene aggiunto il **punteggio del task**, relativa al raggiungimento dell'obbiettivo introdotto nel' `FixedUpdate`, producendo una ricompensa cumulativa, assegnata all'agente:

$$r_t^{\text{cumulative}} = \lambda_I \, r_t^{\text{imitation}} + \lambda_T \, r_t^{\text{task}}$$

Infine, se il task risulta completato con successo, l'episodio viene concluso e ne viene avviato uno nuovo altrimenti si aggiorna il contatore dei frame e si richiede una nuova decisione all'agente.

Algorithm 1 FixedUpdate

```
1: procedure FixedUpdate
2:     if UnwantedObjTouched(spine2, head) then
3:         AddReward(LinkTouchedUnwantedObj())
4:         EndEpisode
5:         return
6:     end if
7:     currentCom ← ComputeComPosition(bodyPartsList)
8:     if TaskIsComplete(r_hand, l_hand) then
9:         taskReward ← LinkTouchedTarget
10:        taskComplete ← True
11:    else
12:        reward ← HandPositionRelationTarget(r_hand, l_hand, target)
13:        taskReward ← reward
14:    end if
15: end procedure
```

Algorithm 2 Update

```
1: procedure Update
2:     if currentFrame ≥ motionLength then
3:         EpisodeInterrupted
4:     end if
5:
6:     currentSceneData ← GetCurrentSceneData
7:     currentAnimData ← GetCurrentAnimationData
8:
9:     poseRw ← PoseRewardCalculator(cPose, tPose)
10:    velRw ← VelocityRewardCalculator(cVelocitie, tVelocitie)
11:    comRw ← ComDeviationRewardCalculator(cCom, tCom)
12:
13:    imitRw ← ImitationRewardCalculator(poseRw, velRew, comRew)
14:    comulativeRw ← ComulativeRewardValue(imitRw, taskReward)
15:    AddReward(comulativeRw)
16:    if taskComplete then
17:        EpisodeInterrupted
18:    end if
19:    currentFrame ← currentFrame + 1
20:    RequestDecision
21: end procedure
```

3.3.2 JointDriveController

La classe `JointDriveController` permette di gestire le giunture articolari del manichino umanoide simulato. È progettata per controllare i movimenti fisici e coordinare l'interazione delle parti del corpo con l'ambiente in Unity. La sua principale responsabilità è configurare e regolare i giunti del manichino, rendendoli responsivi alle azioni generate dall'agente durante l'addestramento. La classe utilizza due principali strutture dati:

- **bodyPartsDict**: Un dizionario che associa ogni Transform di ogni giunto a un oggetto della classe `BodyPart`, contenente tutte le informazioni necessarie per il controllo della giuntura.
- **bodyPartsList**: Una lista che mantiene un riferimento ordinato a tutte le parti del corpo, utile per iterazioni sequenziali.

Durante la fase di inizializzazione, ciascun giunto dell'agente viene rappresentato da un'istanza della classe `BodyPart`, la quale incapsula un componente `Rigidbody`, un `ConfigurableJoint`, oltre alla posizione e rotazione iniziali e ai dati di contatto con l'ambiente circostante. Al `ConfigurableJoint` associato al giunto vengono assegnati specifici parametri che ne determinano il comportamento dinamico:

- **Max Joint Spring**: la coppia elastica utilizzata da Unity per ruotare il giunto dalla sua posizione attuale verso la posizione di destinazione [83].
- **Max Joint Dampen**: riduce la quantità di coppia elastica della molla in proporzione alla differenza tra la velocità attuale dell'articolazione e la sua velocità target. In questo modo si riduce la velocità di movimento del giunto. Questo permette di smorzare le oscillazioni del giunto che altrimenti si protrarrebbero all'infinito [83].
- **Max Joint Force**: limita la quantità di forza che il giunto può applicare [83].

Tali valori sono assegnati alla proprietà **Slerp Drive** all'interno del componente `ConfigurableJoint`, che controlla la rotazione del giunto lungo tutti gli assi locali mediante una coppia motrice. In particolare, i parametri sopra menzionati corrispondono rispettivamente a **position spring**, **position damper** e **maximum force** del componente `ConfigurableJoint`.

Nel contesto del controllo motorio dell'agente, due metodi risultano fondamentali per la regolazione del comportamento articolare:

- `SetJointTargetRotation`: imposta una rotazione target al giunto, specificando gli angoli desiderati lungo gli assi X,Y e Z.
- `SetJointStrength`: regola la forza applicata dal giunto, scalando il valore fornito dalla rete neurale con i limiti massimi di forza configurati.

Capitolo 4

Risultati ed analisi

L'obiettivo principale di questo progetto consiste nello sviluppo di un sistema in grado di controllare autonomamente il movimento di un manichino umanoide all'interno di un ambiente virtuale simulato. L'implementazione richiede che l'agente non solo segua principi di movimento coerenti con l'ergonomia umana, ma che operi anche nel rispetto delle leggi fisiche imposte dal motore di simulazione.

Sebbene l'obiettivo complessivo non sia stato raggiunto nella sua forma completa, i risultati conseguiti rappresentano una base per futuri sviluppi e approfondimenti. Tra i contributi principali si trovano:

- L'implementazione di un manichino biomeccanicamente plausibile, adatto a simulazioni di natura ergonomica. Il modello umanoide è stato dotato di componenti fisici e articolazioni opportunamente configurate, in modo da replicare in maniera quanto più realistica le dinamiche del corpo umano.

- Lo sviluppo di librerie dedicate alla lettura, interpretazione e gestione dei dati di movimento;

- Realizzazione di un sistema per l'addestramento di un manichino umanoide tramite DRL, utilizzando il framework ML-Agents all'interno dell'ambiente Unity.

Una volta definiti i modelli di rappresentazione del movimento, i meccanismi di gestione dei dati e l'architettura decisionale dell'agente, si è infine proceduto alla fase di addestramento. Nei paragrafi seguenti verranno presentati e analizzati nel dettaglio i risultati ottenuti, con riferimento ai comportamenti appresi, alle metriche di valutazione adottate e alle principali osservazioni critiche.

4.1 Setup Sperimentale

Il sistema è stato implementato utilizzando il toolkit **ML-Agents** 3.0.0 all'interno del motore di simulazione Unity 2022.3.39f1. L'ambiente virtuale sviluppato riproduce uno scenario semplificato e controllato. All'interno della scena, oltre alla presenza dell'agente, è definita una postazione di lavoro, costituita da un banco su cui sono disposti vari oggetti target da raggiungere, come illustrato in Figura 4.1.

Al fine di abilitare l'interazione fisica tra l'agente e l'ambiente, la stazione è stata dotata di `collider` applicati agli oggetti coinvolti. Inoltre, ogni GameObject di interesse è stato opportunamente etichettato mediante l'assegnazione di specifici tag semantici, funzionali alla definizione delle regole comportamentali dell'agente:

- **"wall"**: utilizzato per oggetti non rilevanti ai fini del task, che possono essere toccati liberamente e la cui interazione non influisce sulla policy del modello;
- **"target"**: identifica gli oggetti-obiettivo verso cui l'agente deve orientare il proprio comportamento motorio;
- **"ground"**: definisce il piano di appoggio, considerato uno stato critico, poiché l'agente tende a permanervi e può incontrare difficoltà nell'uscirne in modo autonomo.

Per la rappresentazione dell'agente è stato adottato il modello umanoide **SMPL**, personalizzato con una struttura articolare composta da 15 giunti, come descritto nei capitoli precedenti. Il manichino è stato dotato di componenti fisici, in particolare `Rigidbody`, `Collider` e `ConfigurableJoint`. Per garantire la coerenza con le leggi della fisica proprie dell'ambiente simulato, sono stati definiti e calibrati i seguenti parametri:

- una distribuzione realistica della massa tra i segmenti corporei;
- vincoli articolari che delimitano il range di movimento e la forza massima applicabile da ciascun giunto;
- l'inserimento di componenti elastiche e dissipative su ogni articolazione, per garantire la stabilità dinamica e ridurre eventuali oscillazioni indesiderate.

La definizione della struttura semantica dell'ambiente, unita alla corretta configurazione dell'agente, rappresenta un elemento importante per il funzionamento efficace del modello. Questi due aspetti consentono, da un lato, di distinguere in maniera chiara tra componenti rilevanti e irrilevanti ai fini del task; dall'altro, permettono di gestire in modo appropriato le modalità di interazione dell'agente con l'ambiente circostante.

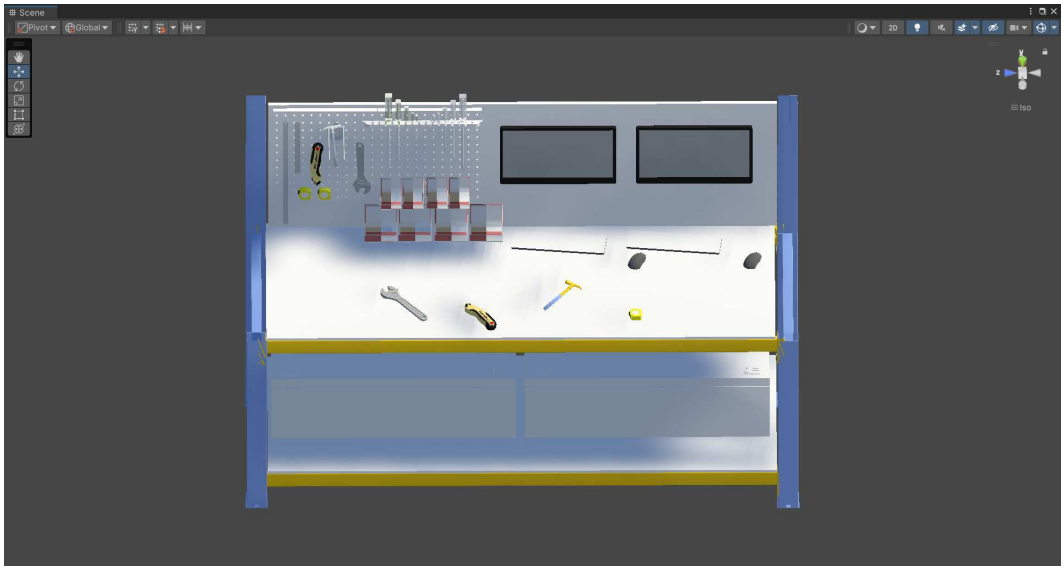


Figura 4.1: Postazione di lavoro utilizzata durante la fase di addestramento del modello. Sulla superficie del tavolo sono stati posizionati diversi oggetti target, che il manichino deve raggiungere con la mano destra al fine di completare il task assegnato.

4.2 Dataset di riferimento

L'animazione da emulare durante la fase di addestramento tramite apprendimento per rinforzo è stata generata mediante un prompt testuale utilizzando il modello pre-addestrato MoMask. Il prompt utilizzato presenta la seguente struttura:

"A person grasping an object above a table with his right hand."

Una volta generata l'animazione, i dati di movimento sono stati esportati in formato BVH e successivamente convertiti in un oggetto di tipo AnimationClip, compatibile con il motore Unity. Tale clip è stata quindi campionata per ottenere una sequenza di frame di riferimento, contenenti le rotazioni locali di ciascun giunto, in accordo con la gerarchia scheletrica del modello. I dati così ottenuti sono stati sottoposti a un processo di preprocessing tramite una libreria sviluppata appositamente. Questo processo ha incluso una ristrutturazione dei dati in una forma gerarchica personalizzata, adeguata all'architettura dell'agente.

A partire da questa rappresentazione, sono state derivate informazioni aggiuntive necessarie all'addestramento, quali le velocità angolari dei giunti e la posizione del centro di massa del corpo. Tali dati costituiscono le principali variabili osservabili per la valutazione dell'imitazione del movimento da parte dell'agente.

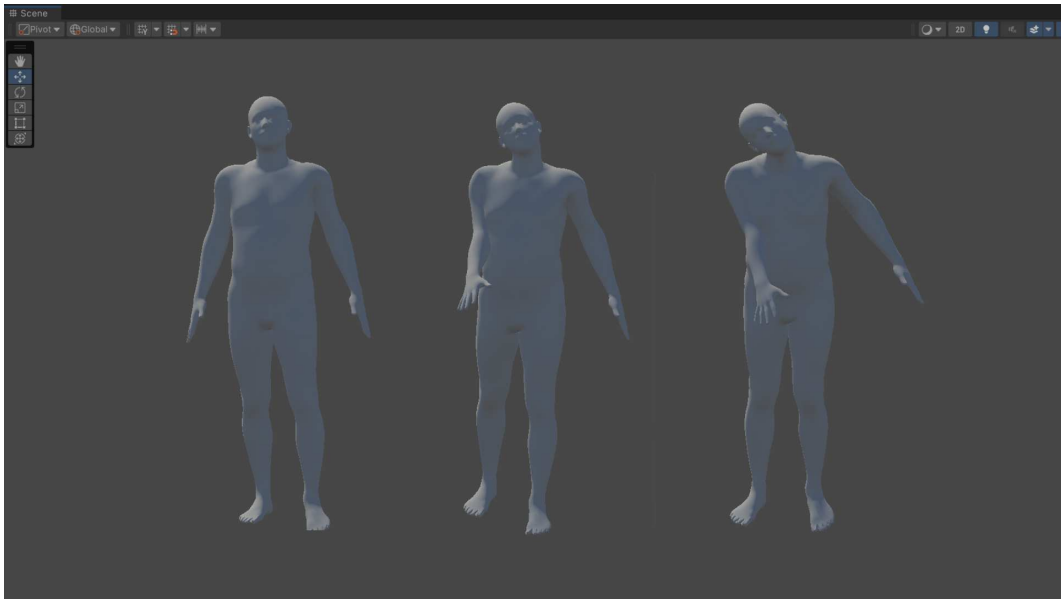


Figura 4.2: Frame dell'animazione generata dal modello MoMask e utilizzata per l'addestramento.

4.3 Parametri di addestramento

Durante la fase di addestramento dell'agente, sono state adottate due configurazioni distinte, differenziate in termini di complessità architetturale e capacità di generalizzazione. L'obiettivo era confrontare l'efficacia di un'impostazione leggera, mirata a una rapida convergenza e all'apprendimento di base dei comportamenti motori, con una configurazione più avanzata, caratterizzata da una rete neurale di maggiore profondità e da strategie esplorative più sofisticate.

In entrambi i casi, è stata adottata una policy di tipo **Proximal Policy Optimization (PPO)**. L'obiettivo comune alle due configurazioni era quello di testare il comportamento dell'agente all'interno dell'ambiente simulato, così da valutare in maniera concreta l'efficacia dell'intero sistema di modellazione e addestramento descritto.

La prima configurazione *Worker1* è stata progettata per minimizzare i tempi di training e consentire un'osservazione rapida dei risultati. Al contrario, la seconda configurazione *Worker2* ha previsto l'impiego di una rete neurale più complessa.

I parametri di addestramento utilizzati per studiare il comportamento del modello, comprendono:

- **Ricompensa cumulativa per step**: questa metrica quantifica la quantità media di ricompensa ottenuta dall'agente per ciascuna azione eseguita nel corso di un episodio. Viene calcolata come il rapporto tra la ricompensa

cumulativa totale accumulata durante l'episodio e la sua durata, espressa in termini di step (ossia il numero complessivo di azioni compiute):

$$\text{reward\_per\_step} = \frac{\text{reward\_per\_episode}}{\text{episode\_length}}$$

Tale misura risulta particolarmente utile nei casi in cui la lunghezza degli episodi non sia costante, ad esempio in task a durata variabile o con possibilità di terminazione anticipata. In questi scenari, la metrica consente di valutare l'efficienza qualitativa del comportamento dell'agente, ovvero la sua capacità di massimizzare la ricompensa per unità di tempo, indipendentemente dalla durata complessiva dell'episodio.

- **Policy loss**: determina quanto varia il processo di decisione dell'agente.
- **Value loss**: rappresenta l'errore medio nella stima del valore degli stati visitati.
- **Entropy**: utilizzata per monitorare il grado di casualità nelle scelte decisionali dell'agente, ovvero il bilanciamento tra esplorazione e sfruttamento.
- **Learning rate**: rappresenta l'intensità con cui l'algoritmo di ottimizzazione aggiorna i parametri del modello a ogni iterazione.

Nei capitoli successivi, le due configurazioni verranno analizzate in modo approfondito e i relativi risultati verranno valutati sulla base dei parametri appena citati, al fine di evidenziare le differenze in termini di efficacia, stabilità e qualità dei comportamenti appresi.

4.3.1 Worker 1

La prima configurazione, definita nel file *Worker1.yaml*, è stata configurata scegliendo gli iperparametri basandosi sulle linee guida ufficiali fornite dalla documentazione di ML-Agents, nonché su configurazioni di riferimento comunemente adottate per task analoghi. I valori selezionati sono riportati nella Tabella 4.1.

Il valore assegnato al batch size, ridotto rispetto agli standard comuni, in combinazione con un buffer size contenuto, consente aggiornamenti più frequenti della rete. Il learning rate è stato impostato a un valore leggermente superiore a quello comunemente utilizzato nei contesti standard, per accelerare la convergenza iniziale. L'architettura della rete neurale impiegata è di tipo compatto, composta da due layer nascosti, ciascuno con un numero limitato di unità.

L'addestramento è stato condotto per un totale di 5.000.000 di step, utilizzando tre agenti in esecuzione parallela su una macchina dotata di processore Intel®

Parametro	**Valore**
Batch size	512
Buffer size	4096
Learning rate	0.0005
Unità nascoste	64
Numero di layer	2
Gamma	0.99
Reward strength	1.0
Orizzonte temporale	500
Max training steps	5.000.000

Tabella 4.1: Parametri utilizzati per l'addestramento dell'agente

Core(TM) i7-8565U CPU @ 1.80GHz (1.99 GHz) e 16 GB di RAM. L'intero processo si è esteso su un arco temporale di circa quattro giorni.

Ricompensa cumulativa per step

Come mostrato in Figura 4.3, si osserva un rapido incremento iniziale della ricompensa per step, che converge rapidamente verso un valore di circa 0.3, indicativo di un apprendimento efficace nelle prime fasi dell'addestramento.

Tale soglia è coerente con la funzione di reward adottata, dove il completamento del task fornisce una ricompensa pari a 1.0, pesata da un coefficiente di 0.3. Il restante contributo, che porta il valore massimo a circa 0.34, è attribuibile all'imitazione del movimento.

Successivamente alla fase iniziale dell'addestramento, la ricompensa per step si mantiene su livelli relativamente stabili, con oscillazioni contenute e un picco massimo registrato nelle fasi conclusive del training. Questo andamento può essere interpretato in due modi complementari: da un lato, potrebbe indicare il raggiungimento di un equilibrio comportamentale stabile da parte dell'agente, capace di mantenere performance costanti nel tempo; dall'altro, il progressivo incremento osservato verso la fine dell'addestramento potrebbe suggerire un'evoluzione del comportamento appreso, culminata in un nuovo plateau prestazionale, potenzialmente associato a una fase di raffinamento della policy.

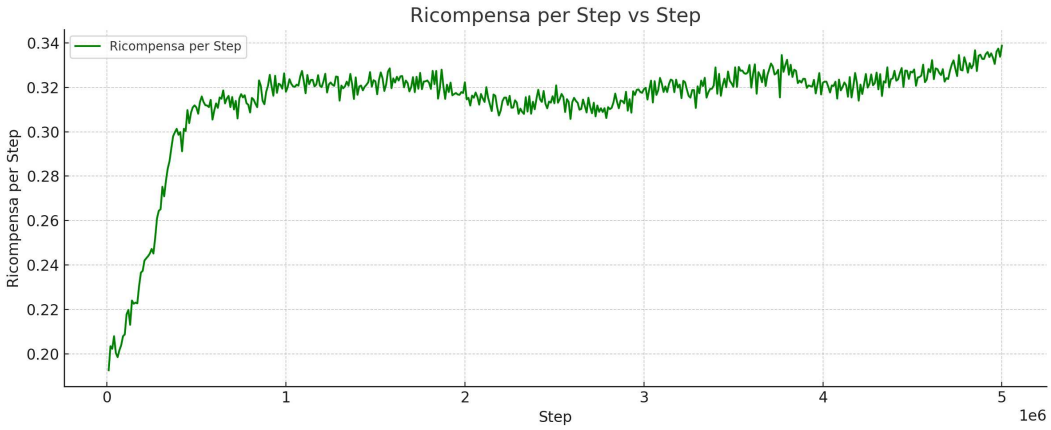


Figura 4.3: Visualizzazione dell'andamento dell'addestramento della ricompensa cumulativa per step Worker1.

Losses/Policy Loss e Value Loss

La **Policy Loss** durante l'addestramento mostra un comportamento molto instabile, suggerendo che l'agente stia cercando di trovare un equilibrio nella sua politica, come rappresentato in Figura 4.4.

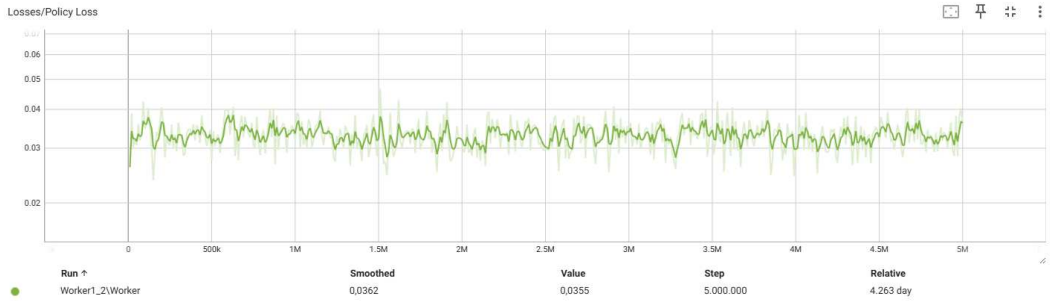


Figura 4.4: Valore della policy loss per l'addestramento con la configurazione worker1.

D'altra parte, la **Value Loss** evidenzia un andamento oscillante ma decrescente nel tempo. Come rappresentato in Figura 4.5, nelle prime fasi dell'addestramento, si osserva un aumento iniziale del valore di loss. Questo è un comportamento atteso, poiché nelle fasi iniziali la policy dell'agente è fortemente casuale e lo porta a esplorare stati molto eterogenei, spesso mai visti prima.

Successivamente, si nota una progressiva diminuzione della Value Loss. Questo indica che il modello inizia a definire con maggiore precisione il valore atteso degli stati, man mano che l'agente esplora l'ambiente e raffina la propria policy. Tuttavia,

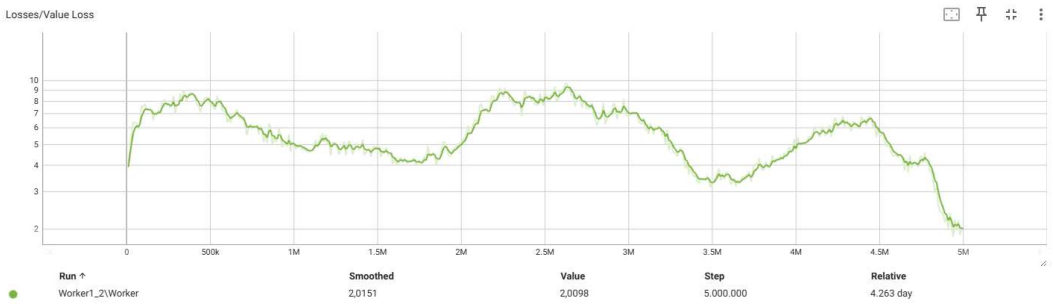


Figura 4.5: Valore della value loss per l'addestramento con la configurazione worker1.

non si osserva una discesa lineare, infatti il grafico mostra alcune fasi in cui la Value Loss torna a salire. Tuttavia, queste ricrescite non rappresentano un fallimento del processo di apprendimento, ma sono un fenomeno naturale nei modelli di reinforcement learning come PPO. Ogni volta che la policy dell'agente cambia in modo significativo, ad esempio perché apprende un nuovo comportamento, il modello si ritrova a dover stimare il valore di stati nuovi o distribuiti diversamente, e quindi può incorrere in errori temporanei più alti. Ciò che risulta essere positivo è che, nonostante queste oscillazioni, il trend complessivo della Value Loss tende a diminuire nel tempo.

Policy/entropy

Come mostrato nella Figura 4.6, l'andamento dell'entropia nel tempo appare relativamente stabile, con modeste oscillazioni attorno a un valore medio costante. Tale comportamento suggerisce che, durante l'intera fase di training, la policy dell'agente ha mantenuto un livello di esplorazione elevato e persistente. In altri termini, l'agente ha continuato a selezionare le azioni in modo non deterministico, senza convergere verso una strategia ottimale ben definita.

Idealmente, ci si attenderebbe una progressiva riduzione dell'entropia man mano che l'agente apprende, segnalando una crescente preferenza per le azioni più vantaggiose e una conseguente diminuzione della componente esplorativa. Tuttavia, il trend attuale implica che, sebbene l'agente sia stato in grado di ottenere ricompense cumulative soddisfacenti, esso non abbia ancora consolidato una policy deterministica, continuando a mantenere un comportamento esplorativo superiore al necessario per ottimizzare la performance in modo efficace.

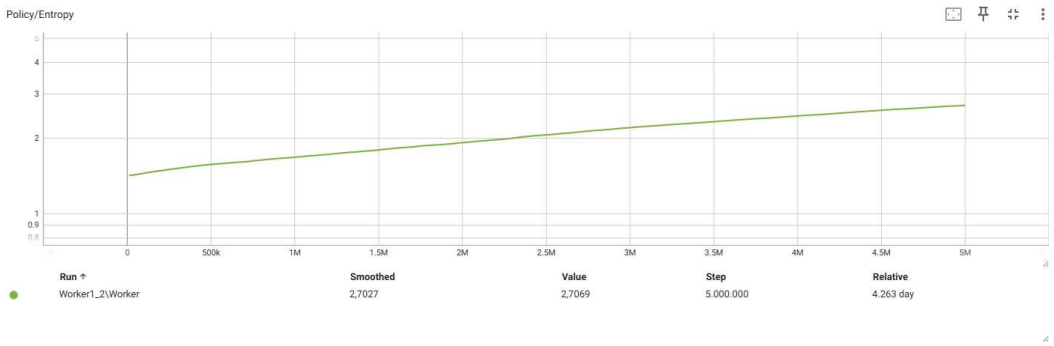


Figura 4.6: Valore dell'entropia per l'addestramento con la configurazione worker1.

Policy/Learning Rate

Nel file di configurazione Worker1, è stato adottato un learning rate costante. Questo approccio, se da un lato assicura una velocità di apprendimento stabile e prevedibile nelle prime fasi del training, può tuttavia risultare subottimale nelle fasi più avanzate, quando la policy è già prossima a una soluzione efficace e sarebbero invece auspicabili aggiornamenti di ampiezza ridotta. Nonostante ciò, nel presente contesto sperimentale si è deliberatamente optato per un learning rate costante, in quanto tale scelta si è rivelata funzionale all'obiettivo specifico di osservare con maggiore chiarezza l'evoluzione del comportamento dell'agente in condizioni controllate.

4.3.2 Worker 2

Nel file di configurazione *Worker2.yaml*, la complessità della rete neurale utilizzata per l'addestramento è stata incrementata, al fine di favorire una maggiore capacità di rappresentazione e generalizzazione del comportamento dell'agente. I valori selezionati per i principali iperparametri sono riportati nella Tabella 4.2.

Rispetto alla configurazione precedente, sia il batch size che il buffer size sono stati aumentati, consentendo l'aggiornamento dei parametri su un numero più ampio di esperienze accumulate. Il learning rate è stato aggiornato su un valore standard, ed è stato impiegato un scheduler dinamico, optando per un tasso di apprendimento lineare. L'architettura della rete neurale, pur mantenendosi relativamente compatta, è stata estesa ma i layer nascosti sono rimasti invariati.

L'addestramento è stato condotto per un totale di 5.000.000 di step, utilizzando trentacinque agenti in esecuzione parallela su una workstation HP Z4 G4 dotata di processore Intel® Xeon® W-2223 @ 3.60GHz, 32GB di RAM e una GPU dedicata NVIDIA Quadro P2200 con 5 GB di VRAM. L'intero processo si è esteso su un arco temporale di circa due giorni.

Parametro	**Valore**
Batch size	1024
Buffer size	10240
Learning rate	0.0003
Unità nascoste	128
Numero di layer	2
Gamma	0.99
Reward strength	1.0
Orizzonte temporale	500
Max training steps	5.000.000

Tabella 4.2: Parametri utilizzati per l'addestramento dell'agente

Ricompensa cumulativa per step

L'andamento della reward per step evidenzia un picco marcato attorno ai 2 milioni e 4.5 milioni di step, seguito da una fase di regressione dei valori. Tale comportamento può essere ricondotto a diversi fattori, tra cui un cambiamento improvviso nella policy appresa, una riduzione dell'efficacia esplorativa o una temporanea instabilità nel processo di ottimizzazione. È plausibile ipotizzare che l'agente, dopo aver raggiunto una strategia particolarmente efficiente, abbia avviato una fase di esplorazione alternativa, abbandonando temporaneamente la soluzione precedentemente consolidata.

Nella parte finale dell'addestramento, l'analisi della reward per step mostra tuttavia un trend generalmente positivo, con una progressiva risalita dei valori medi. Questo comportamento lascia ipotizzare che l'agente fosse ancora in fase di ottimizzazione della propria policy, e che un'estensione del numero di step avrebbe potuto consolidare ulteriormente tale progresso, portando potenzialmente a performance più elevate. Inoltre, risulta evidente come il valore massimo raggiunto in questa configurazione superi significativamente il picco massimo ottenuto con la configurazione Worker1, rappresentato dalla linea tratteggiata rossa in Figura 4.7. Tale superamento conferma l'efficacia della configurazione nell'esprimere policy più performanti.

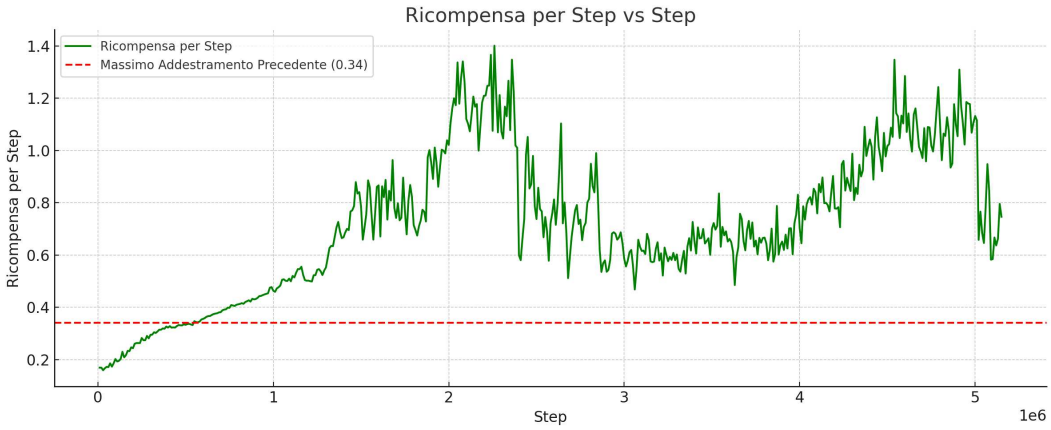


Figura 4.7: Visualizzazione dell'andamento dell'addestramento della ricompensa cumulativa per step Worker2.

Losses/Policy Loss e Value Loss

L'andamento delle metriche di loss conferma quanto osservato nella reward per step. La Value Loss si riduce rapidamente e si stabilizza su valori bassi, come mostrato in Figura 4.8, indicando che il critico ha appreso efficacemente a stimare i valori degli stati. Parallelamente, la Policy Loss mostrata in Figura 4.9, cresce graduale fino a circa 2 milioni di step, momento in cui si verifica anche un picco nella reward per step. Questo suggerisce che la policy stesse evolvendo verso un comportamento più efficiente, ma ancora instabile. La successiva diminuzione della reward per step, accompagnata da una Policy Loss costante ma elevata, indica che la nuova strategia appresa non si è ancora consolidata.

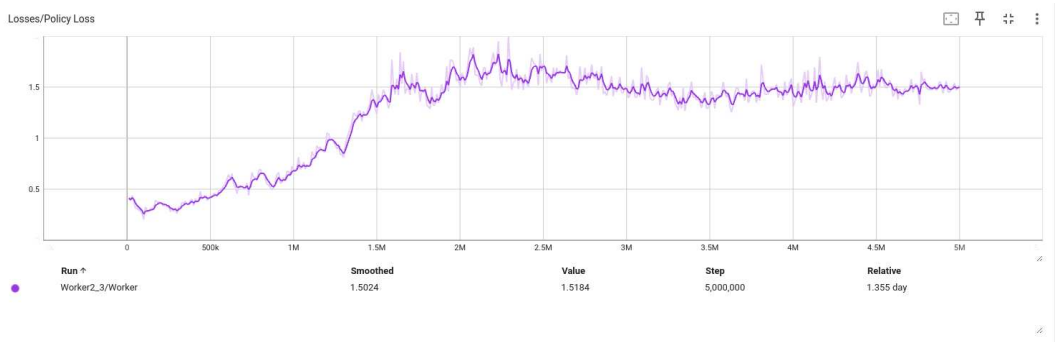


Figura 4.8: Valore della policy loss per l'addestramento con la configurazione worker2.

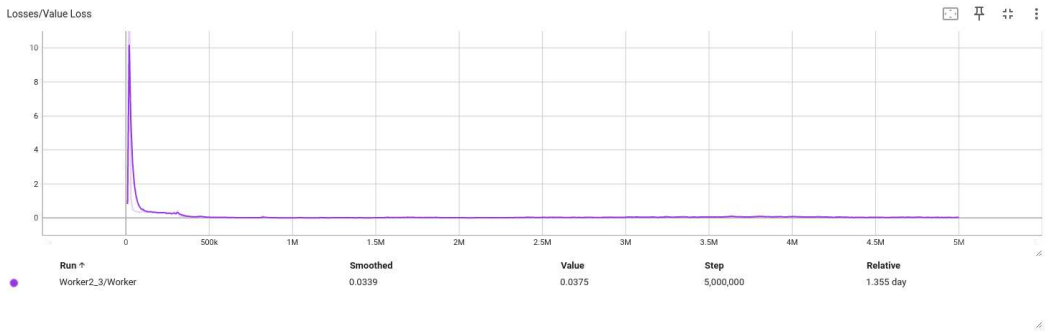


Figura 4.9: Valore della value loss per l'addestramento con la configurazione worker2.

Policy/entropy

L'entropia della policy mostra in Figura 4.10 un andamento decrescente costante, passando da circa 1.45 a 0.63 lungo l'intero processo di addestramento. Questo suggerisce che l'agente ha gradualmente ridotto la componente esplorativa del proprio comportamento, consolidando scelte sempre più deterministiche. La discesa lenta e continua dell'entropia è coerente con un apprendimento progressivo, senza collassi prematuri della policy. In particolare, il valore di entropia registrato attorno ai 2 milioni di step coincide con il picco nella ricompensa per step, indicando che la policy stava diventando più mirata ed efficace.

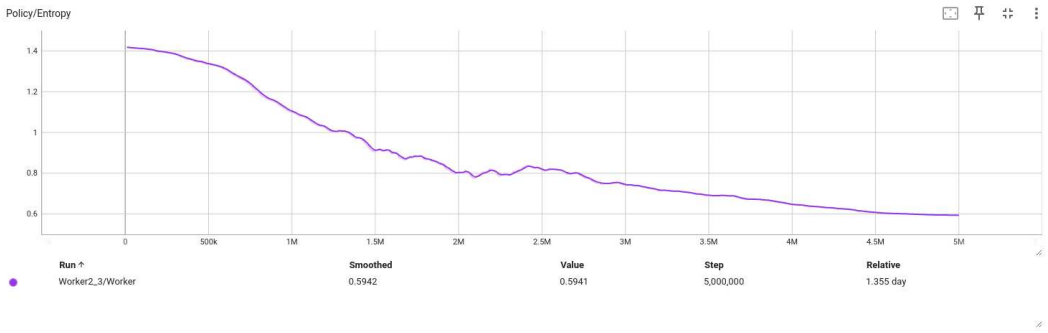


Figura 4.10: Valore dell'entropia per l'addestramento con la configurazione worker2.

Policy/Learning Rate

Il learning rate associato alla policy è stato regolato attraverso una schedulazione lineare decrescente, come specificato nel file di configurazione. Questa strategia

ha consentito un apprendimento più rapido nelle fasi iniziali dell'addestramento, seguito da una progressiva fase di consolidamento della policy nelle fasi avanzate. Tale andamento risulta coerente con la riduzione dell'entropia e con la stabilizzazione della policy loss, come evidenziato nei grafici presentati in precedenza.

Tuttavia, la diminuzione della reward per step successiva al picco massimo potrebbe suggerire che, nelle fasi finali, un learning rate eccessivamente ridotto abbia limitato la capacità del modello di adattarsi a strategie più vantaggiose. In altre parole, l'agente potrebbe aver perso flessibilità nel migliorare ulteriormente la propria policy, rimanendo ancorato a comportamenti subottimali. L'andamento di questa metrica è illustrato in Figura 4.11, dove è possibile osservare l'effetto della schedulazione sulla dinamica dell'apprendimento.

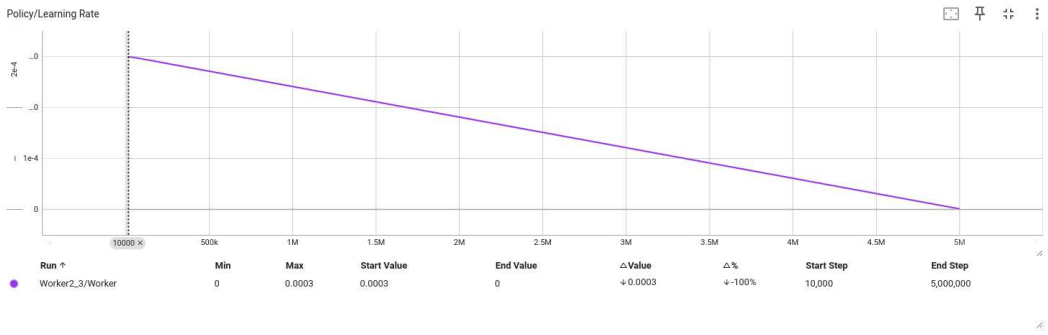


Figura 4.11: Valore dell'entropia per l'addestramento con la configurazione worker2.

4.3.3 Confronto fra le configurazioni di addestramento

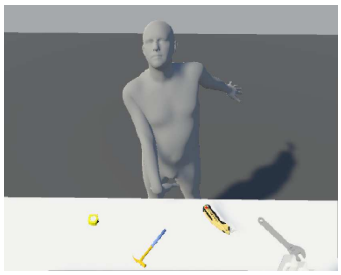
Gli esperimenti condotti con le configurazioni *Worker1.yaml* e *Worker2.yaml* hanno permesso di analizzare l'impatto della complessità architetturale e della parametrizzazione dell'addestramento sulle prestazioni dell'agente. Entrambi gli addestramenti si sono svolti per 5.000.000 di step, ma con differenze nelle risorse computazionali, nella struttura della rete neurale e nella gestione degli iperparametri.

Worker1, caratterizzato da una rete compatta e parametri conservativi, ha mostrato un apprendimento rapido nelle fasi iniziali, con una ricompensa per step che si è stabilizzata attorno a 0.3. L'agente ha mantenuto un comportamento stabile ma fortemente esplorativo, come evidenziato dal valore di entropia crescente nel tempo. La **Value Loss** ha mostrato un trend decrescente ma con fisiologiche oscillazioni, mentre la **Policy Loss** è rimasta instabile, riflettendo un continuo tentativo di miglioramento della policy.

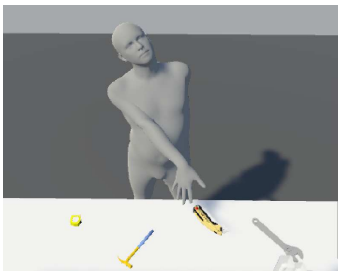
Worker2, invece, ha utilizzato una rete più ampia e un setup computazionale più avanzato. L'agente ha raggiunto un picco di ricompensa per step significativamente

più alto rispetto a Worker1, indicando una maggiore capacità di apprendere policy più efficaci. Tuttavia, questo picco è stato seguito da una fase di regressione, probabilmente dovuta a un cambiamento strategico nella policy. Nelle fasi finali, la ricompensa ha ripreso a salire, suggerendo che l'agente fosse ancora in fase di ottimizzazione. La **Value Loss** si è rapidamente stabilizzata su valori bassi, mentre la **Policy Loss** è cresciuta fino al picco di performance, mantenendosi poi elevata. L'entropia ha mostrato una discesa graduale, evidenziando una transizione verso una policy più deterministica e consolidata.

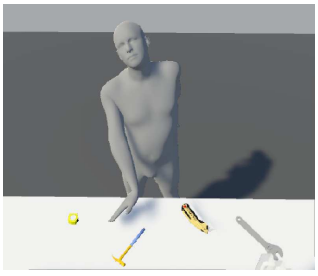
È importante sottolineare che, nonostante i risultati incoraggianti, entrambi gli addestramenti presentano una natura ancora approssimativa. I 5.000.000 di step utilizzati non sono sufficienti per valutare con completezza l'effettivo andamento a lungo termine della rete neurale, né per garantire la piena convergenza della policy. Un numero maggiore di iterazioni sarebbe necessario per osservare la stabilizzazione finale del comportamento dell'agente, il consolidamento delle strategie apprese. A seguire, in Figura 4.12 viene presentata un'analisi sintetica dell'andamento dell'addestramento per l'intero arco dei 5 milioni di step.



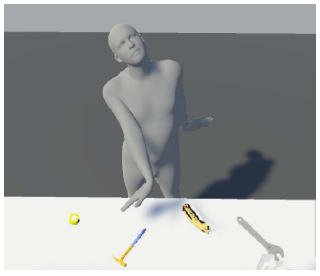
(a) 1 milione di step



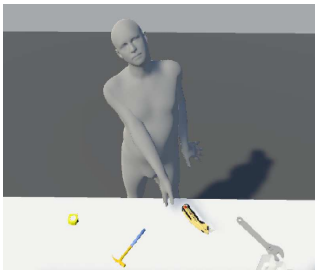
(b) 2 milioni di step



(c) 3 milioni di step



(d) 4 milioni di step



(e) 5 milioni di step

Figura 4.12: Sequenza di immagini che mostra l'evoluzione del comportamento del manichino umanoide con la configurazione Worker2.yaml per Reinforcement Learning. Si osserva che all'aumenmtare degli step, un progressivo miglioramento nella coordinazione e nella direzione del movimento verso l'oggetto target posto sul tavolo e del braccio non adibito al task.

Capitolo 5

Conclusioni e Prospettive Future

L'obiettivo di questa tesi era sviluppare un sistema in grado di generare movimenti umani realistici e coerenti con le leggi fisiche, all'interno di ambienti virtuali di lavoro, sfruttando tecniche di intelligenza artificiale e simulazione biomeccanica. Il progetto si colloca all'intersezione tra simulazione fisica, apprendimento per rinforzo e studio dell'ergonomia, con l'ambizione di creare un agente umanoide in grado di apprendere movimenti naturali partendo da dati reali.

Nonostante l'obiettivo finale non sia stato completamente raggiunto, il lavoro svolto ha permesso di costruire le fondamenta su cui tale risultato potrà essere conseguito in futuro.

Durante la realizzazione del lavoro, sono state sviluppate:

- una pipeline per il parsing e l'interpretazione di file BVH contenenti dati di movimento umano;
- un sistema di agenti fisici completamente controllati tramite rete neurale, con una struttura biomeccanica personalizzata;
- una libreria di supporto per la gestione dei dati di addestramento;
- una logica di reward progettata per guidare il comportamento dell'agente verso l'imitazione dei movimenti desiderati.

Un ulteriore punto di forza del progetto è stata l'adozione del motore di gioco **Unity** in combinazione con il framework **ML-Agents**, che ha consentito di integrare facilmente simulazione fisica, animazione e apprendimento automatico all'interno di un unico ambiente coerente. L'uso di Unity ha garantito una rappresentazione visiva immediata e altamente personalizzabile, utile sia in fase di debugging sia per

l'analisi ergonomica dei movimenti. ML-Agents, invece, ha offerto un'infrastruttura modulare e accessibile per l'addestramento degli agenti, con supporto a tecniche avanzate di reinforcement learning. Questa scelta tecnologica rende il sistema sviluppato facilmente estendibile, riutilizzabile e adattabile a scenari futuri più complessi.

Gli esperimenti di addestramento, condotti sulle due diverse configurazioni di addestramento, hanno mostrato segnali positivi nella capacità dell'agente di recepire le ricompense e apprendere parzialmente alcune dinamiche motorie. Tuttavia, le reti non sono riuscite a convergere verso policy stabili ed efficaci nel riprodurre i movimenti target. Le cause principali sono da ricercarsi nei limiti computazionali, nei tempi di addestramento ridotti e nella complessità intrinseca del task proposto.

In particolare, le metriche raccolte mostrano che la rete era ancora in fase di ottimizzazione al termine dei 5 milioni di step, e che un numero maggiore di iterazioni — supportato da un'infrastruttura hardware più potente — sarebbe stato necessario per raggiungere la convergenza.

Nonostante ciò, il lavoro risulta comunque significativo per il contributo metodologico e tecnico che apporta: l'ambiente di simulazione creato, la struttura del manichino fisico e le logiche di interazione tra movimento e reward possono essere riutilizzati e potenziati in futuri studi.

I principali sviluppi futuri possibili includono:

- il proseguimento dell'addestramento con maggiori risorse computazionali;
- il perfezionamento della funzione di ricompensa per favorire una convergenza più rapida e stabile;
- l'introduzione di tecniche di curriculum learning o hybrid imitation + reinforcement learning;
- la validazione dei risultati tramite confronti quantitativi con indici ergonomici.

In conclusione, pur non avendo raggiunto completamente l'obiettivo prestazionale previsto, il progetto ha prodotto strumenti, metodi e conoscenze utili, rappresentando un punto di partenza concreto per la realizzazione futura di sistemi di simulazione ergonomica basati su intelligenza artificiale e ambienti virtuali in Unity.

Bibliografia

[1] Ministero del Lavoro e delle Politiche Sociali. *Testo Unico sulla Salute e Sicurezza sul Lavoro, D.Lgs. 81/08*. Accessed: October 16, 2024. Giu. 2016. URL: https://www.lavoro.gov.it/documenti-e-norme/studi-e-statistiche/Documents/Testo%20Unico%20sulla%20Salute%20e%20Sicurezza%20sul%20Lavoro/Testo-Unico-81-08-Edizione-Giugno%202016.pdf (cit. a p. i).

[2] Zohaib Jan, Farhad Ahamed, Wolfgang Mayer, Niki Patel, Georg Grossmann, Markus Stumptner e Ana Kuusk. «Artificial intelligence for industry 4.0: Systematic review of applications, challenges, and opportunities». In: *Expert Systems with Applications* 216 (2023), p. 119456. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.119456. URL: https://www.sciencedirect.com/science/article/pii/S0957417422024757 (cit. a p. 2).

[3] Ercan Oztemel e Samet Gursev. «Literature review of Industry 4.0 and related technologies». In: *Journal of Intelligent Manufacturing* 31.1 (2020), pp. 127–182. ISSN: 1572-8145. DOI: 10.1007/s10845-018-1433-8. URL: https://doi.org/10.1007/s10845-018-1433-8 (cit. a p. 2).

[4] Maulshree Singh, Evert Fuenmayor, Eoin P. Hinchy, Yuansong Qiao, Niall Murray e Declan Devine. «Digital Twin: Origin to Future». In: *Applied System Innovation* 4.2 (2021). ISSN: 2571-5577. DOI: 10.3390/asi4020036. URL: https://www.mdpi.com/2571-5577/4/2/36 (cit. a p. 4).

[5] Shirine El Zaatari, Mohamed Marei, Weidong Li e Zahid Usman. «Cobot programming for collaborative industrial tasks: An overview». In: *Robotics and Autonomous Systems* 116 (2019), pp. 162–180. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2019.03.003. URL: https://www.sciencedirect.com/science/article/pii/S092188901830602X (cit. a p. 4).

[6] Tiago Zonta, Cristiano André Da Costa, Rodrigo da Rosa Righi, Miromar Jose de Lima, Eduardo Silveira da Trindade e Guann Pyng Li. «Predictive maintenance in the Industry 4.0: A systematic literature review». In: *Computers & Industrial Engineering* 150 (2020), p. 106889 (cit. a p. 4).

[7] Detlef Zuehlke. «SmartFactory—Towards a factory-of-things». In: *Annual reviews in control* 34.1 (2010), pp. 129–138 (cit. a p. 4).

[8] Robert S. Bridger. *Introduction to Ergonomics*. 3rd. CRC Press, 2008. DOI: 10.1201/9781439894927. URL: https://doi.org/10.1201/9781439894927 (cit. alle pp. 5–7).

[9] Zamir J. Espíritu, Karen M. Canaza e Luis C. Rada. «Systematic Review of Ergonomic Risks in Musculoskeletal Disorders of Workers in an Industrial Company». In: *4th LACCEI International Multiconference on Entrepreneurship, Innovation and Regional Development - LEIRD 2024*. Virtual Edition: LACCEI, 2024, pp. 1–8. DOI: 10.18687/LEIRD2024.1.1.324. URL: https://dx.doi.org/10.18687/LEIRD2024.1.1.324 (cit. a p. 5).

[10] NC State Ergonomics Center. *ErgoHowl Newsletter - Q2 2021*. Accessed: 2024-11-02. 2021. URL: https://ergocenter.ncsu.edu/ergohowl_q2_2021/ (cit. a p. 5).

[11] Francesco Marcolin, Gianna Mian, Adriano Ossicini, Fernando Luisi, Sergio Pischiottin e Liza Vecchi Brumatti. *Glossario di Ergonomia*. A cura di Giuseppe Cimaglia. In collaborazione con ERGOLAB, laboratorio di usabilità e ricerca ergonomica. Sovraintendenza Medica Generale, 2024 (cit. alle pp. 5–7).

[12] Marcelo M. Soares e Francisco Rebelo. *Ergonomics in Design: Methods & Techniques*. Boca Raton, London, New York: CRC Press, Taylor & Francis Group, 2017. ISBN: 978-1-4987-6070-6. URL: http://www.crcpress.com (cit. a p. 5).

[13] Interaction Design Foundation. *Human-Centered Design - Topic Overview*. Accessed: 2024-11-02. 2024. URL: https://www.interaction-design.org/literature/topics/human-centered-design (cit. a p. 6).

[14] Arto Reiman, Jari Kaivo-oja, Elina Parviainen, Esa-Pekka Takala e Theresa Lauraeus. «Human factors and ergonomics in manufacturing in the industry 4.0 context – A scoping review». In: *Technology in Society* 65 (2021), p. 101572. ISSN: 0160-791X. DOI: https://doi.org/10.1016/j.techsoc.2021.101572. URL: https://www.sciencedirect.com/science/article/pii/S0160791X21000476 (cit. alle pp. 6–8).

[15] Elena Stefana, Filippo Marciano, Diana Rossi, Paola Cocca e Giuseppe Tomasoni. «Wearable Devices for Ergonomics: A Systematic Literature Review». In: *Sensors* 21.3 (2021). ISSN: 1424-8220. DOI: 10.3390/s21030777. URL: https://www.mdpi.com/1424-8220/21/3/777 (cit. alle pp. 6–8, 10).

[16] Luca Gualtieri, Erwin Rauch e Renato Vidoni. «Emerging research fields in safety and ergonomics in industrial collaborative robotics: A systematic literature review». In: *Robotics and Computer-Integrated Manufacturing* 67 (2021), p. 101998. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2020.101998. URL: https://www.sciencedirect.com/science/article/pii/S073658452030209X (cit. alle pp. 6, 8).

[17] Damian Grajewski, Filip Górski, Przemysław Zawadzki e Adam Hamrol. «Application of Virtual Reality Techniques in Design of Ergonomic Manufacturing Workplaces». In: *Procedia Computer Science* 25 (2013). 2013 International Conference on Virtual and Augmented Reality in Education, pp. 289–301. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2013.11.035. URL: https://www.sciencedirect.com/science/article/pii/S1877050913012404 (cit. alle pp. 6, 8).

[18] Víctor Igelmo, Anna Syberfeldt, Dan Högberg, Francisco García Rivera e Estela Pérez Luque. «Aiding observational ergonomic evaluation methods using MOCAP systems supported by AI-based posture recognition». In: *DHM2020*. IOS Press, 2020, pp. 419–429 (cit. alle pp. 7, 8).

[19] John R. Wilson. «Virtual environments applications and applied ergonomics». In: *Applied Ergonomics* 30.1 (1999), pp. 3–9. ISSN: 0003-6870. DOI: https://doi.org/10.1016/S0003-6870(98)00040-4. URL: https://www.sciencedirect.com/science/article/pii/S0003687098000404 (cit. alle pp. 7, 8, 10).

[20] Stefania Tagliafierro. «Design and development of a postural prediction system for Digital Human Model in 3D virtual work-station assessments». Tesi di dott. Politecnico di Torino, 2022 (cit. alle pp. 9, 14, 15, 20, 21).

[21] Automazione News. *CIM4.0 porta a MECSPE Additive Manufacturing ed Extended Reality*. Accessed: 2024-11-02. 2024. URL: https://www.automazionenews.it/cim4-0-porta-a-mecspe-additive-manufacturing-ed-extended-reality/ (cit. a p. 10).

[22] Google Cloud. *Deep Learning vs. Machine Learning: What's the Difference?* Accessed: 2024-11-02. 2024. URL: https://cloud.google.com/discover/deep-learning-vs-machine-learning (cit. a p. 11).

[23] Universe IT. *Apprendimento Supervisionato e Non Supervisionato*. Accessed: 2024-11-02. 2024. URL: https://universeit.blog/apprendimento-supervisionato-e-non-supervisionato/ (cit. a p. 11).

[24] Wikipedia contributors. *Apprendimento per rinforzo*. Accessed: 2024-11-02. 2024. URL: https://it.wikipedia.org/wiki/Apprendimento_per_rinforzo (cit. a p. 11).

[25] International Organization for Standardization (ISO). *Artificial Intelligence and Machine Learning Standards*. Accessed: 2024-11-02. 2024. URL: https://www.iso.org/artificial-intelligence/machine-learning (cit. a p. 11).

[26] Victor C.H. Chan, Gwyneth B. Ross, Allison L. Clouthier, Steven L. Fischer e Ryan B. Graham. «The role of machine learning in the primary prevention of work-related musculoskeletal disorders: A scoping review». In: *Applied Ergonomics* 98 (2022), p. 103574. DOI: 10.1016/j.apergo.2021.103574. URL: https://doi.org/10.1016/j.apergo.2021.103574 (cit. alle pp. 12, 20).

[27] Occupational Health & Safety (OH&S). *How AI-Driven Algorithms Improve an Individual's Ergonomic Safety*. Accessed: 2024-11-02. 2020. URL: https://ohsonline.com/articles/2020/05/14/how-aidriven-algorithms-improve-an-individuals-ergonomic-safety.aspx (cit. a p. 12).

[28] Srimantha E. Mudiyanselage, Phuong Hoang Dat Nguyen, Mohammad Sadra Rajabi e Reza Akhavian. «Automated Workers' Ergonomic Risk Assessment in Manual Material Handling Using sEMG Wearable Sensors and Machine Learning». In: *Electronics* 10 (2021). DOI: 10.3390/electronics10202558. URL: https://doi.org/10.3390/electronics10202558 (cit. a p. 12).

[29] Nicolò Massobrio. «Realtà Virtuale per l'addestramento alla gestione del rischio idrogeologico. Creazione di uno scenario virtuale e gestione degli NPC.» Tesi di laurea mag. Politecnico di Torino, 2020. URL: https://webthesis.biblio.polito.it/15325/1/tesi.pdf (cit. a p. 15).

[30] Visbox, Inc. *CAVE - Virtual Reality Immersive System*. Accessed: 2024-11-02. 2024. URL: https://www.visbox.com/products/cave/ (cit. a p. 16).

[31] HTC Vive. *VIVE - Virtual Reality System*. Accessed: 2024-11-02. 2024. URL: https://www.vive.com/us/ (cit. a p. 16).

[32] Meta. *Meta Quest - Virtual Reality Headsets*. Accessed: 2024-11-02. 2024. URL: https://www.meta.com/it/quest/ (cit. a p. 16).

[33] Unity Technologies. *Unity: Real-time Development Platform*. https://unity.com/. Accessed: October 15, 2024 (cit. a p. 17).

[34] IUDAV. *Unity: Un motore di gioco potente e versatile*. 2024. URL: https://www.iudav.it/tecnologie/unity-un-motore-di-gioco-potente-e-versatile/ (cit. a p. 17).

[35] Multiplayer.it. *Unreal Engine 5: Tecnologie, Feature e Futuro*. 2024. URL: https://multiplayer.it/articoli/unreal-engine-5-tecnologie-feature-futuro.html (cit. a p. 17).

[36] HTML.it. *Guida a Unreal Engine*. 2024. URL: https://www.html.it/guide/unreal-engine/ (cit. a p. 18).

[37] Develop4Fun. *Requisiti ottimali per eseguire Unreal Engine 5 in modo fluido su Windows e macOS*. 2024. URL: https://www.develop4fun.it/requisiti-ottimali-per-eseguire-unreal-engine-5-in-modo-fluido-su-windows-e-macos/ (cit. a p. 18).

[38] Gavriel Salvendy e Waldemar Karwowski. *Handbook of Human Factors and Ergonomics, Fifth Edition*. John Wiley & Sons, Inc., 2021 (cit. a p. 19).

[39] Wenmin Zhu, Xiumin Fan e Yanxin Zhang. «Applications and research trends of digital human models in the manufacturing industry». In: *Virtual Reality & Intelligent Hardware* 1.6 (2019), pp. 558–579. ISSN: 2096-5796. DOI: 10.1016/j.vrih.2019.09.005. URL: https://www.sciencedirect.com/science/article/pii/S2096579619300828 (cit. a p. 19).

[40] Patricia Marcos, T. Seitz, Heiner Bubb, Andreas Wichert e H. Feussner. «Computer simulation for ergonomic improvements in laparoscopic surgery». In: *Applied Ergonomics* 37 (2006), pp. 251–258. DOI: 10.1016/j.apergo.2005.09.003 (cit. a p. 19).

[41] Giorgio Colombo, Stefano Filippi, Paolo Rissone e Caterina Rizzi. «ICT Methodologies to Model and Simulate Parts of Human Body for Prosthesis Design». In: *Proceedings of the International Conference on Product Lifecycle Management (PLM)*. 2007, pp. 559–568. DOI: 10.1007/978-3-540-73321-8_64 (cit. a p. 19).

[42] Lars Hanson, Dan Högberg, Daniel Lundström e Maria Wårell. «Application of Human Modelling in Health Care Industry». In: *Proceedings of the International Conference on Digital Human Modeling*. 2009, pp. 521–530. ISBN: 978-3-642-02808-3. DOI: 10.1007/978-3-642-02809-0_55 (cit. a p. 19).

[43] Christian Vogt, Christian Mergl e Heiner Bubb. «Interior layout design of passenger vehicles with RAMSIS». In: *Human Factors and Ergonomics in Manufacturing & Service Industries* 15 (2005), pp. 197–212. DOI: 10.1002/hfm.20022 (cit. a p. 19).

[44] Thorsten Kuebler, Hans Wirsching e David Barnes. *Ramsis-Digital Human Modeling for Optimized Safety and Survivability of the Warfighter*. 2018 (cit. a p. 19).

[45] Wikipedia contributors. *Jack (human modeling) — Wikipedia, The Free Encyclopedia*. Accessed: 2024-11-02. 2020. URL: https://en.wikipedia.org/wiki/Jack_(human_modeling) (cit. a p. 19).

[46] The University of Michigan. *3D Static Strength Prediction Program*. 2011 (cit. a p. 19).

[47] Chika Edith Mgbemena, Ashutosh Tiwari, Yuchun Xu, Vinayak Prabhu e Windo Hutabarat. «Ergonomic evaluation on the manufacturing shop floor: A review of hardware and software technologies». In: *CIRP Journal of Manufacturing Science and Technology* 30 (2020), pp. 68–78. ISSN: 1755-5817. DOI: 10.1016/j.cirpj.2020.04.003. URL: https://www.sciencedirect.com/science/article/pii/S1755581720300316 (cit. a p. 19).

[48] R. Castellone, Stefania Spada, Giovanni Caiazzo e Maria Pia Cavatorta. «Assessment of Anthropometric Differences in the Design of Workstations: Case Studies of an Automotive Assembly Line». In: *Proceedings of an International Conference on Ergonomics and Human Factors*. 2017 (cit. a p. 19).

[49] Matthew P. Reed, Sarah M. Ebert, Christian D'Souza e Matthew L. H. Jones. «Predicting Standing Reach Postures using Deep Neural Networks». In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 66.1 (2022), pp. 662–666. DOI: 10.1177/1071181322661150. URL: https://doi.org/10.1177/1071181322661150 (cit. a p. 20).

[50] B. Zhang, I. Horváth, J.F.M. Molenbroek e C. Snijders. «Using artificial neural networks for human body posture prediction». In: *International Journal of Industrial Ergonomics* 40.4 (2010), pp. 414–424. ISSN: 0169-8141. DOI: 10.1016/j.ergon.2010.02.003. URL: https://www.sciencedirect.com/science/article/pii/S0169814110000284 (cit. a p. 20).

[51] Mohammad Moe Bataineh, R. Marler e Karim Abdel-Malek. «Artificial Neural Network-Based Prediction of Human Posture». In: lug. 2013, pp. 305–313. ISBN: 978-3-642-39181-1. DOI: 10.1007/978-3-642-39182-8_36 (cit. a p. 20).

[52] Francesco Pistolesi, Michele Baldassini e Beatrice Lazzerini. «A human-centric system combining smartwatch and LiDAR data to assess the risk of musculoskeletal disorders and improve ergonomics of Industry 5.0 manufacturing workers». In: *Computers in Industry* 155 (2024), p. 104042. ISSN: 0166-3615. DOI: 10.1016/j.compind.2023.104042. URL: https://www.sciencedirect.com/science/article/pii/S0166361523001926 (cit. a p. 20).

[53] Viktor Losing Taizo Yoshikawa e Emel Demircan. «Machine learning for human movement understanding». In: *Advanced Robotics* 34.13 (2020), pp. 828–844. DOI: 10.1080/01691864.2020.1786724. eprint: https://doi.org/10.1080/01691864.2020.1786724. URL: https://doi.org/10.1080/01691864.2020.1786724 (cit. a p. 20).

[54] Mattias Billast, Jonas De Bruyne, Klaas Bombeke, Tom De Schepper e Kevin Mets. «Physical ergonomics anticipation with human motion prediction». In: SciTePress. 2024 (cit. alle pp. 20, 21).

[55] Kedi Lyu, Haipeng Chen, Zhenguang Liu, Beiqi Zhang e Ruili Wang. «3D human motion prediction: A survey». In: *Neurocomputing* 489 (2022), pp. 345–365. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.02.045. URL: https://www.sciencedirect.com/science/article/pii/S0925231222002077 (cit. a p. 20).

[56] Sebastian Starke, He Zhang, Taku Komura e Jun Saito. «Neural state machine for character-scene interactions». In: *ACM Transactions on Graphics* 38.6 (2019), p. 178 (cit. alle pp. 20, 21).

[57] Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee e Jehee Lee. «Learning predict-and-simulate policies from unorganized human motion data». In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–11 (cit. alle pp. 20, 21).

[58] Sanjay Nambiar, Anton Wiberg e Mehdi Tarkian. «Automation of unstructured production environment by applying reinforcement learning». In: *Frontiers in Manufacturing Technology*. 2023. URL: https://api.semanticscholar.org/CorpusID:257602917 (cit. a p. 21).

[59] Xue Bin Peng, Pieter Abbeel, Sergey Levine e Michiel Van de Panne. «Deepmimic: Example-guided deep reinforcement learning of physics-based character skills». In: *ACM Transactions On Graphics (TOG)* 37.4 (2018), pp. 1–14 (cit. a p. 21).

[60] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll e Michael J. Black. «AMASS: Archive of Motion Capture as Surface Shapes». In: *International Conference on Computer Vision*. Ott. 2019, pp. 5442–5451 (cit. alle pp. 21, 22, 32).

[61] Catalin Ionescu, Dragos Papava, Vlad Olaru e Cristian Sminchisescu. «Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (2014), pp. 1325–1339. DOI: 10.1109/TPAMI.2013.248. URL: http://vision.imar.ro/human3.6m (cit. a p. 22).

[62] Leonid Sigal, Alexandru O. Balan e Michael J. Black. «HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion». In: *International Journal of Computer Vision* 87 (2010), pp. 4–27. DOI: 10.1007/s11263-009-0273-6. URL: https://doi.org/10.1007/s11263-009-0273-6 (cit. a p. 22).

[63] Chuan Guo, Xinxin Zuo, Sen Wang, Shihao Zou, Qingyao Sun, Annan Deng, Minglun Gong e Li Cheng. «Action2motion: Conditioned generation of 3d human motions». In: *Proceedings of the 28th ACM International Conference on Multimedia*. 2020, pp. 2021–2029 (cit. alle pp. 22, 32).

[64] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll e Michael J. Black. «SMPL: A Skinned Multi-Person Linear Model». In: *ACM Transactions on Graphics (TOG)* 34.6 (2015), 248:1–248:16. DOI: 10.1145/2816795.2818013 (cit. a p. 23).

[65] Kathleen Robinette, Susan Blackwell, Hein Daanen, Marc Boehmer, Scott Fleming, Teresa Brill, Dietrich Hoeferlin e Donna Burnside. *Civilian American and European Surface Anthropometry Resource (CAESAR) final report*. Tech. Rep. AFRL-HEWP-TR-2002-0169. US Air Force Research Laboratory, 2002 (cit. a p. 24).

[66] Meredith S. Maddock. *Motion Capture File Formats Explained*. Rapp. tecn. Accessed: October 16, 2024. Department of Computer Science, University of Sheffield, 2023. URL: http://www.dcs.shef.ac.uk/~mikem/mocapfileformats.pdf (cit. a p. 24).

[67] Xin Chen, Biao Jiang, Wen Liu, Zilong Huang, Bin Fu, Tao Chen e Gang Yu. «Executing your Commands via Motion Diffusion in Latent Space». In: *arXiv preprint arXiv:2212.04048* (2023). URL: https://arxiv.org/abs/2212.04048 (cit. alle pp. 25, 26).

[68] Chuan Guo, Yuxuan Mu, Muhammad Gohar Javed, Sen Wang e Li Cheng. «MoMask: Generative Masked Modeling of 3D Human Motions». In: (2023). Code available at https://github.com/EricGuo5513/momask-codes. arXiv: 2312.00063 [cs.CV]. URL: https://ericguo5513.github.io/momask/ (cit. alle pp. 25, 26, 31).

[69] Nhat M. Hoang, Kehong Gong, Chuan Guo e Michael Bi Mi. «MotionMix: Weakly-Supervised Diffusion for Controllable Motion Generation». In: *arXiv preprint arXiv:2401.11115* (2024). URL: https://arxiv.org/abs/2401.11115 (cit. a p. 25).

[70] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li e Li Cheng. «Generating Diverse and Natural 3D Human Motions from Text». In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 5142–5151 (cit. a p. 26).

[71] Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee e Jehee Lee. «Learning Predict-and-Simulate Policies From Unorganized Human Motion Data». In: *ACM Trans. Graph.* 38.6 (2019). Code available at https://github.com/snumrl/ICC?tab=readme-ov-file. URL: https://mrl.snu.ac.kr/publications/ProjectICC/ICC.html (cit. a p. 26).

[72] Xue Bin Peng, Pieter Abbeel, Sergey Levine e Michiel van de Panne. «DeepMimic: Example-guided Deep Reinforcement Learning of Physics-based Character Skills». In: *ACM Trans. Graph.* 37.4 (lug. 2018). Code available at `https://github.com/xbpeng/DeepMimic`, 143:1–143:14. ISSN: 0730-0301. DOI: `10.1145/3197517.3201311`. URL: `http://doi.acm.org/10.1145/3197517.3201311` (cit. alle pp. 26, 28, 29, 36, 47, 49).

[73] Andrea Tirinzoni, Ahmed Touati, Jesse Farebrother, Mateusz Guzek, Anssi Kanervisto, Yingchen Xu, Alessandro Lazaric e Matteo Pirotta. «Zero-shot Whole-Body Humanoid Control via Behavioral Foundation Models». In: (2024). Code available at `https://github.com/facebookresearch/metamotivo`. URL: `https://ai.meta.com/research/publications/zero-shot-whole-body-humanoid-control-via-behavioral-foundation-models/` (cit. alle pp. 27, 28).

[74] Zhi Cen, Huaijin Pi, Sida Peng, Zehong Shen, Minghui Yang, Zhu Shuai, Hujun Bao e Xiaowei Zhou. «Generating Human Motion in 3D Scenes from Text Descriptions». In: *CVPR*. Code available at `https://github.com/zju3dv/text_scene_motion`. 2024. URL: `https://drive.google.com/file/d/1ZqJdvObz87CwuOlmUnYZNei54tieMZpZ/view` (cit. a p. 27).

[75] Jeremy Kress. *RagdollTrainer - The Evolution of Active Ragdoll Simulation in Unity*.
urlhttps://github.com/kressdev/RagdollTrainer/tree/main. 2021 (cit. a p. 29).

[76] Unity Technologies. *ML-Agents - Learning Environment Examples: Walker*. 2025. URL: `https://github.com/Unity-Technologies/ml-agents/blob/develop/docs/Learning-Environment-Examples.md#walker` (cit. a p. 29).

[77] Matthias Plappert, Christian Mandery e Tamim Asfour. «The KIT Motion-Language Dataset». In: *Big Data* 4.4 (dic. 2016), pp. 236–252. DOI: `10.1089/big.2016.0028`. URL: `http://dx.doi.org/10.1089/big.2016.0028` (cit. a p. 32).

[78] Unity Technologies. *ML-Agents Overview: Model Types*. `https://unity-technologies.github.io/ml-agents/ML-Agents-Overview/#model-types`. Accessed: October 16, 2024 (cit. a p. 33).

[79] Vibhav Chitale. *Unity ML Tutorial*. `https://vibhavchitale.weebly.com/unity-ml-tutorial.html`. Accessed: October 16, 2024 (cit. a p. 34).

[80] Emiliana VT. *BVH Tools - GitHub Repository*. Accessed: 2024-11-02. 2024. URL: `https://github.com/emilianavt/BVHTools` (cit. a p. 38).

[81] Stanislav Zhuk. *Ragdoll Helper*. `https://github.com/StasClick/ragdoll_helper`. Accessed: November 2, 2024 (cit. alle pp. 41, 44).

[82] Max Planck Institute for Intelligent Systems. *SMPL - A Skinned Multi-Person Linear Model*. `https://smpl.is.tue.mpg.de/download.php`. Accessed: November 2, 2024 (cit. a p. 41).

[83] Unity Technologies. *ConfigurableJoint*. URL: `https://docs.unity3d.com/Manual/class-ConfigurableJoint.html` (cit. a p. 54).