

Deep Reinforcement Learning per la Simulazione Pedonale: Analisi di Scenari ad Alta Densità

Relatore: Professore Giuseppe Vizzari

Co-relatore: Dottoressa Daniela Briola

Relazione della prova finale di:

Andrea Falbo

Matricola 887525

Anno Accademico 2023/2024

Indice

Introduzione	1
1 Stato dell'arte	3
1.1 Modellazione e simulazione di pedoni	3
1.2 Agenti Intelligenti	4
1.2.1 Classi di agenti intelligenti	5
1.3 Interazione con ambiente	7
1.4 Reinforcement Learning	9
1.4.1 Introduzione	9
1.4.2 Deep Reinforcement Learning	11
1.4.3 Curriculum Learning	12
1.4.4 PPO	13
1.5 Valutazione modelli e validazione	14
2 Un modello di DRL per la simulazione di pedoni	16
2.1 Strumenti	16
2.2 Glossario	17
2.3 Ambiente	19
2.4 Agente	20
2.4.1 Raggi	21
2.4.2 Archi	22
2.4.3 Spazio delle azioni	22
2.4.4 Osservazioni	23
2.5 Rewards	26
2.6 Curriculum	28
2.7 Addestramento	34
2.7.1 Early Fail	35
2.7.2 Fasi di Addestramento	35
2.7.3 Risultati addestramento	36
2.8 Test	38
2.8.1 Ambienti di Test	38
2.8.2 Analisi dei risultati	40
3 Descrizione del progetto software	43
3.1 Componenti Godot	43

3.1.1	Training Scene	43
3.1.2	Level Batch	43
3.1.3	Sync	44
3.1.4	Level Manager	44
3.1.5	Pedestrian	44
3.2	Componenti Python	45
3.2.1	Runner	45
3.2.2	Config	45
3.2.3	Callbacks	47
3.3	Flusso di esecuzione	48
3.3.1	Training Scene	48
3.3.2	Level Batch	49
3.3.3	Level Manager	50
4	Analisi ad Alta Densità	52
4.1	Introduzione	52
4.2	Glossario	52
4.3	Casi di studio	55
4.3.1	Analisi della larghezza del corridoio prima di un bottleneck	55
4.3.2	Confronto dei tempi di evacuazione di collegamenti centrali e ad angolo	56
4.4	Modello ad alta densità	56
4.4.1	Test su modello base	56
4.4.2	Ambienti	58
4.4.3	Rewards	61
4.5	Andamento dell’addestramento	61
4.6	Test	62
4.6.1	Analisi dei risultati	65
4.7	Analisi di Sensitività	73
4.7.1	Introduzione	73
4.7.2	Configurazioni	75
4.7.3	Risultati	76
5	Conclusione e sviluppi futuri	83
Bibliografia		86

Introduzione

Le *simulazioni di pedoni e folle* stanno acquisendo un crescente interesse in molteplici settori, dalla pianificazione urbanistica alla gestione delle emergenze. Queste simulazioni sono fondamentali per supportare la progettazione di spazi pubblici, prevedere le dinamiche dei pedoni in luoghi affollati e simulare evacuazioni in situazioni di emergenza. La realizzazione di sistemi per la simulazione pedonale è complessa e richiede uno studio approfondito delle dinamiche pedonali e dei vari approcci modellistici disponibili.

Tradizionalmente, i modelli di simulazione pedonale si basano su *algoritmi definiti manualmente* sulla base di teorie in merito al movimento delle persone. Tuttavia, questa relazione esplora un approccio innovativo e ancora poco esplorato: il *Reinforcement Learning* (RL), una branca del *Machine Learning* (ML) per l'addestramento di agenti autonomi.

L'obiettivo principale della relazione è sviluppare *agenti intelligenti* utilizzando il Reinforcement Learning per simulare realisticamente il comportamento dei pedoni, replicando fedelmente le loro azioni con limitazioni simili a quelle umane, come la visione limitata e capacità decisionali ristrette.

Per la creazione degli ambienti virtuali è stato utilizzato *Godot Engine*, con la gestione degli agenti di Reinforcement Learning tramite *Godot-RL-Agents* e l'implementazione degli algoritmi di RL con la libreria *Stable Baselines 3* in PyTorch. Inoltre, è stata impiegata la libreria *PedPy* per l'analisi del movimento pedonale.

Il modello ottenuto offre prestazioni comparabili a quelle dei modelli definiti manualmente e dimostra la capacità di generalizzare quanto appreso durante l'addestramento per risolvere scenari inediti.

In seguito al conseguimento dei comportamenti desiderati e di risultati realistici, si è deciso di analizzare *scenari ad alta densità*. La seconda parte del progetto si è quindi incentrata sull'addestramento di agenti in grado di operare efficacemente in condizioni di traffico intenso, preservando tutti i comportamenti realistici già ottenuti.

La relazione è organizzata come segue:

- Nel Capitolo 1 viene presentata una panoramica attuale sulla ricerca nel campo della simulazione pedonale e del Reinforcement Learning.
- Nel Capitolo 2 viene presentato un modello di Reinforcement Learning per la simulazione di pedoni.
- Nel Capitolo 3 viene descritta l'architettura software, in particolare i metodi e le tecniche utilizzate per la realizzazione del progetto.
- Nel Capitolo 4 viene esteso il modello base analizzando delle situazioni ad alta densità.
- Nel Capitolo 5 viene effettuata una considerazione finale sul lavoro eseguito. La relazione si conclude con alcuni possibili sviluppi futuri.

Capitolo 1

Stato dell'arte

1.1 Modellazione e simulazione di pedoni

Gli approcci alla simulazione dei pedoni e delle folle hanno storicamente adottato un paradigma basato su *agenti* [3], ma ciò non implica necessariamente una rilevanza diretta per il settore dell'intelligenza artificiale (IA). I modelli di pedoni e folle sono stati prevalentemente fondati su approcci fisici, come il modello delle forze sociali, sugli automi cellulari, che variano da quelli più semplici, come [6], a quelli più complessi, come [4] e su modelli più propriamente definiti da agenti autonomi [9]. Gli obiettivi di tali simulazioni sono spesso orientati verso il *realismo visivo*, senza necessariamente perseguire una validazione rigorosa basata su misurazioni di flussi, densità e velocità [17] in una vasta gamma di situazioni.

Un primo tentativo di integrare tecniche di apprendimento automatico nel modellare il comportamento dei pedoni ha utilizzato metodi di apprendimento per rinforzo (Q-learning) e approcci di classificazione [10], come gli alberi decisionali, per scegliere tra un insieme limitato di azioni di movimento disponibili, basandosi sulla percezione della situazione corrente.

Recentemente, diversi studi hanno esplorato l'uso di tecniche di *regressione* per prevedere il vettore di velocità del pedone [2] o prevedere sia la velocità del pedone che la direzione [11]. Ad esempio, alcuni lavori hanno utilizzato reti neurali profonde addestrate con dati di tracciamento dei pedoni ottenuti da filmati di esperimenti , dove la velocità adottata nel fotogramma successivo del video era considerata come la verità di riferimento. Tuttavia, questo approccio presenta limitazioni simili a quelle delle tecniche di previsione delle traiettorie, come un orizzonte di previsione ridotto e una generalizzazione limitata dei modelli ottenuti, che possono riprodurre fedelmente situazioni simili a quelle del set di addestramento, ma risultare meno rappresentativi in contesti diversi.

Più di recente, l'approccio di apprendimento per rinforzo [12] è stato ulteriormente sviluppato, come dimostrato da studi in cui gli autori hanno definito un modello di percezione che fornisce all'agente informazioni su un insieme finito di agenti vicini, sull'ostacolo più vicino e sull'obiettivo finale. Il modello di azione regola il vettore di velocità dell'agente in termini di variazione dell'angolo e accelerazione/decelerazione.

Il modello discusso in questo contesto è stato significativamente ispirato da tali approcci, ma con la differenza che, invece di eseguire un processo di addestramento specifico per ciascun esperimento, è stato definito un unico processo di addestramento basato su un approccio curriculare. Questa metodologia presenta situazioni che dovrebbero insegnare all'agente competenze di base, garantendo così la possibilità di ottenere risultati plausibili in un'ampia gamma di situazioni. L'*apprendimento curriculare*, infatti, è un approccio generale ma, nel contesto dell'apprendimento per rinforzo, è considerato un promettente metodo di *transfer learning* per ottenere un alto livello di generalizzazione [7] nei modelli comportamentali degli agenti.

1.2 Agenti Intelligenti

Per studiare il comportamento dei pedoni, viene utilizzato il modello dell'agente intelligente. Questo modello è definito come un'entità capace di percepire l'ambiente circostante attraverso sensori e di eseguire azioni tramite attuatori [16]. La Figura 1.1 illustra schematicamente le componenti principali di un agente e le interazioni di base con l'ambiente, rappresentato come l'insieme di tutto ciò che l'agente può percepire.

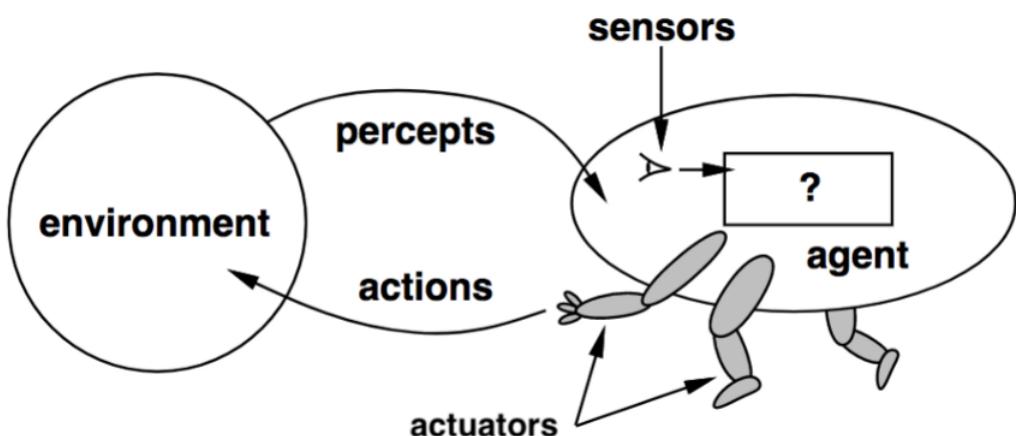


Figura 1.1: Diagramma di un semplice agente intelligente

Gli agenti, nonostante condividano una struttura e un funzionamento di base comune, possono differire enormemente per complessità e capacità. Una classificazione possibile li suddivide in 5 classi in base al grado di intelligenza e capacità di percezione.

1.2.1 Classi di agenti intelligenti

Di seguito viene fornita una breve descrizione di ciascuna classe di agente secondo la categorizzazione di Russell e Norvig, utilizzando gli schemi presenti nel libro [16].

Agenti a riflessi semplici Questi sono gli agenti più semplici e agiscono esclusivamente sulla base della percezione corrente. La condizione necessaria per il funzionamento di questi agenti è la completa osservabilità dell'ambiente.

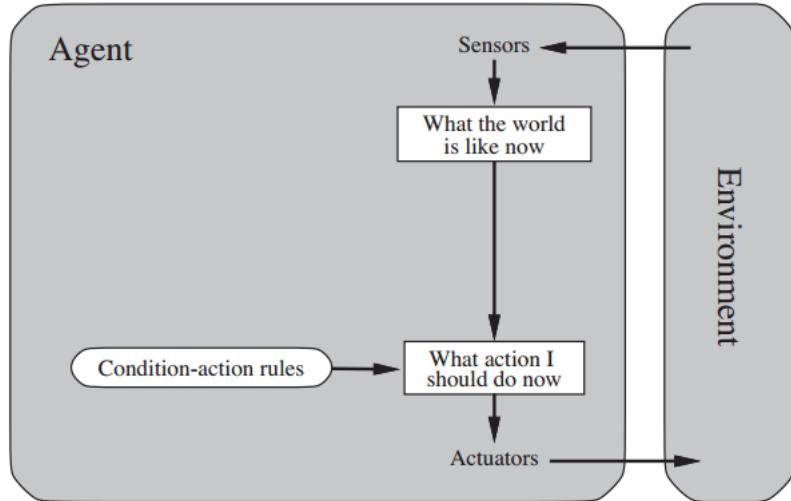


Figura 1.2: Schema di un agente a riflessi semplici tratta da [16]

Agenti a riflessi basati su modello A differenza degli agenti a riflessi semplici, questi possono operare in ambienti parzialmente osservabili. La caratteristica distintiva è la presenza di un modello interno che include uno stato corrente. Questo stato dipende da uno storico delle percezioni e da una serie di regole interne.

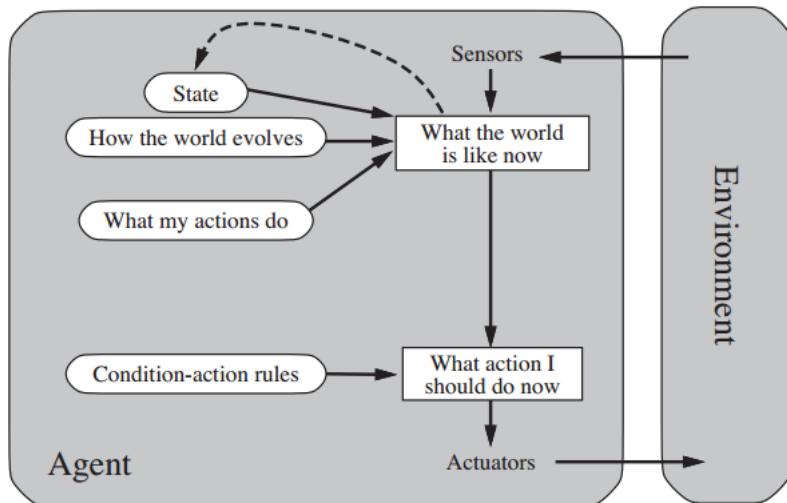


Figura 1.3: Schema di un agente a riflessi basato su modello tratta da [16]

Agenti guidati da obiettivo Gli agenti guidati da obiettivo si basano su un modello e includono una descrizione degli stati obiettivo e di quelli da evitare. Il modello distingue quindi gli stati desiderabili da quelli non desiderabili.

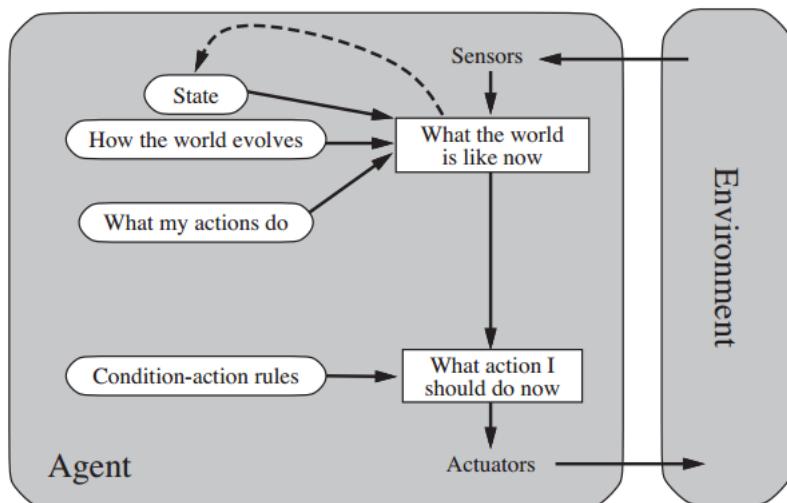


Figura 1.4: Schema di un agente guidato a obiettivi tratta da [16]

Agenti guidati da utility L’evoluzione naturale del modello guidato da obiettivo è il modello basato su utility, che supera la visione binaria della desiderabilità introducendo una metrica. Un agente basato sulla misura di utility ha come obiettivo la massimizzazione di questo valore.

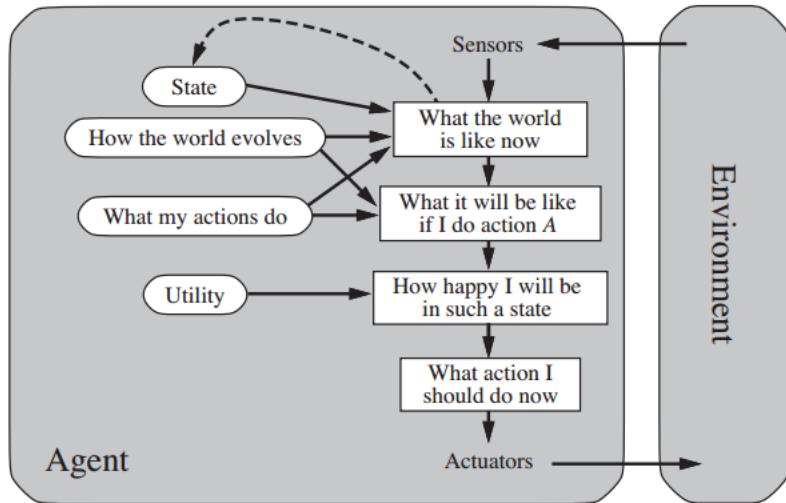


Figura 1.5: Schema di un agente guidato da utility tratta da [16]

Agenti capaci di apprendimento La massima espressione degli agenti intelligenti è rappresentata dagli agenti capaci di apprendere. Questi agenti iniziano da un ambiente completamente sconosciuto e, nel tempo, apprendono i comportamenti migliori in base alle risposte dell'ambiente e alla stima della performance attuale dell'agente.

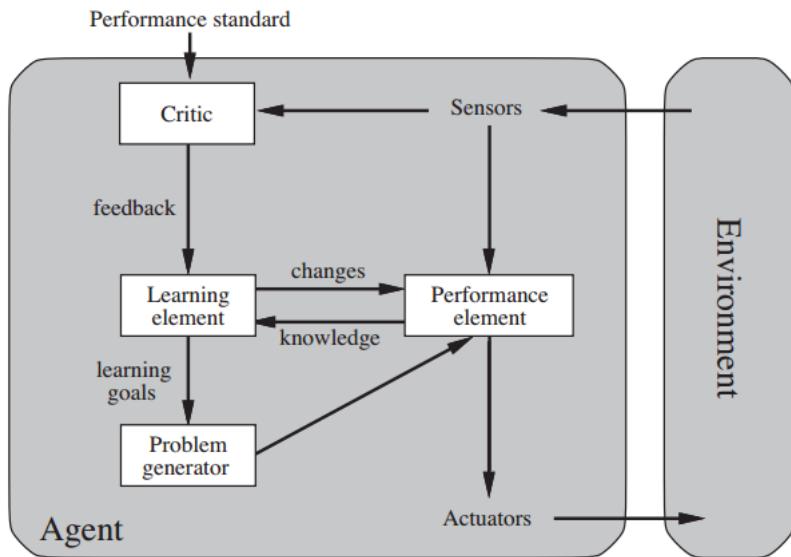


Figura 1.6: Schema di un agente capace di apprendimento tratta da [16]

1.3 Interazione con ambiente

Le interazioni tra agente e ambiente possono essere di diversa natura e rappresentano un elemento cruciale per la suddivisione delle classi di agenti. Una prima importante distinzione è tra modelli continui e discreti. Nei modelli continui, le variabili di spazio, tempo e stato sono continue, il che au-

menta inevitabilmente la complessità di gestione rispetto ai modelli discreti, rendendo talvolta impossibile l'implementazione. I modelli discreti, invece, accettano una certa perdita di informazioni in cambio di miglioramenti in termini di prestazioni e semplicità del modello. La discretizzazione può essere applicata a una o più variabili, con quella dello spazio tra le più utilizzate, poiché semplifica notevolmente i processi decisionali dell'agente e l'evoluzione dell'ambiente. Inoltre, facilita il calcolo delle statistiche sugli agenti nei modelli microscopici. In Figura 1.7, possiamo osservare un ambiente con spazio e azioni discreto.¹



Figura 1.7: Scacchi è un esempio di ambiente con spazio discretizzato

Quando si elabora un modello, è fondamentale definire come l'agente percepisce l'ambiente. Integrare tutte le percezioni sensoriali umane è spesso troppo complesso a livello computazionale, poiché oltre ai cinque sensi, bisogna considerare anche le componenti cognitive e di pianificazione. Nei modelli odierni, la *percezione* dell'agente si basa spesso sulla vista e sulle informazioni che può estrapolare dagli oggetti presenti nell'ambiente, come marker o indicazioni che possono contenere informazioni, o anche da altri agenti. Ad esempio, in [14], viene presentato un modello in cui la percezione dell'ambiente da parte dell'agente è basata esclusivamente sulla vista e tutte le altre informazioni derivano da interpretazioni di ciò che è stato osservato.

L'ambiente, quindi, diventa un fornitore di informazioni che ogni agente

¹<https://www.chess.com/>

può interpretare in modo diverso, proprio come avviene nella realtà. Anche il tempo è una variabile che può essere discretizzata e utilizzata per estrapolare informazioni o scandire la rapidità con cui l'agente compie determinate azioni e prende decisioni. L'attivazione degli agenti e la scelta dell'azione successiva possono essere gestite temporalmente in modi diversi: alcuni modelli potrebbero richiedere sincronismo tra gli agenti, attivando decisioni e azioni a determinati momenti temporali, mentre altri potrebbero permettere maggiore indipendenza, con agenti che prendono decisioni autonomamente ogni delta di tempo. In generale, discretizzare il tempo, solitamente tramite l'unità base dei *timesteps*, semplifica notevolmente le operazioni e rende il modello più snello.

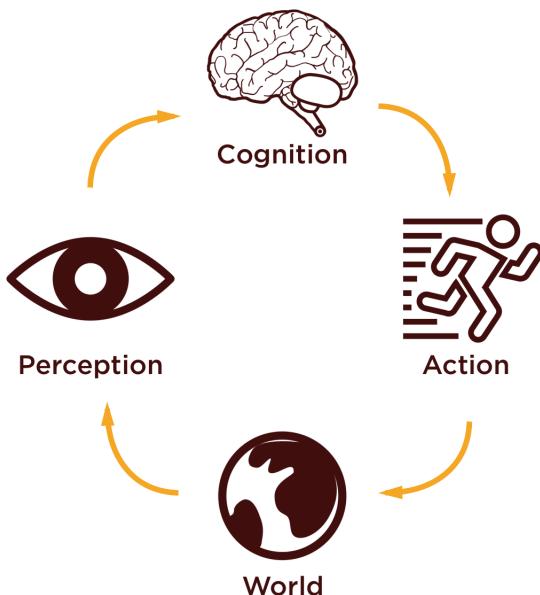


Figura 1.8: Ciclo di percezione, decisione e azione

1.4 Reinforcement Learning

1.4.1 Introduzione

L'*intelligenza artificiale* (IA) è un ramo dell'informatica che si occupa della creazione di programmi informatici capaci di dimostrare intelligenza. Tradizionalmente, qualsiasi software che mostri abilità cognitive come percezione, ricerca, pianificazione e apprendimento è considerato parte dell'IA.

Tutti i programmi informatici che dimostrano intelligenza sono considerati IA, ma non tutti gli esempi di IA possono apprendere. Il *machine learning* (ML) è l'area dell'IA che si occupa della creazione di programmi informatici

in grado di risolvere problemi che richiedono intelligenza, apprendendo dai dati. Ci sono tre principali rami del ML: apprendimento supervisionato, apprendimento non supervisionato e apprendimento per rinforzo.

- L'*apprendimento supervisionato* (SL) ha il compito di apprendere da dati etichettati. In SL, un umano decide quali dati raccogliere e come etichettarli. L'obiettivo del SL è generalizzare.
- L'*apprendimento non supervisionato* (UL) ha il compito di apprendere da dati non etichettati. Anche se i dati non necessitano più di essere etichettati, i metodi utilizzati dal computer per raccogliere i dati devono comunque essere progettati da un umano. L'obiettivo in UL è comprimere.
- L'*apprendimento per rinforzo* (RL) ha il compito di apprendere attraverso il *trial-and-error*. In questo tipo di compito, nessun umano etichetta i dati e nessun umano raccoglie o progetta esplicitamente la raccolta dei dati. L'obiettivo in RL è agire.

Un recente e potente approccio all'apprendimento automatico, denominato deep learning (DL), si basa sull'uso di approssimazioni di funzioni non lineari multi-strato, comunemente note come reti neurali. Il deep learning rappresenta un insieme di tecniche e metodologie avanzate che impiegano reti neurali per affrontare e risolvere una vasta gamma di compiti di machine learning, inclusi l'apprendimento supervisionato (SL), l'apprendimento non supervisionato (UL) e l'apprendimento per rinforzo (RL).

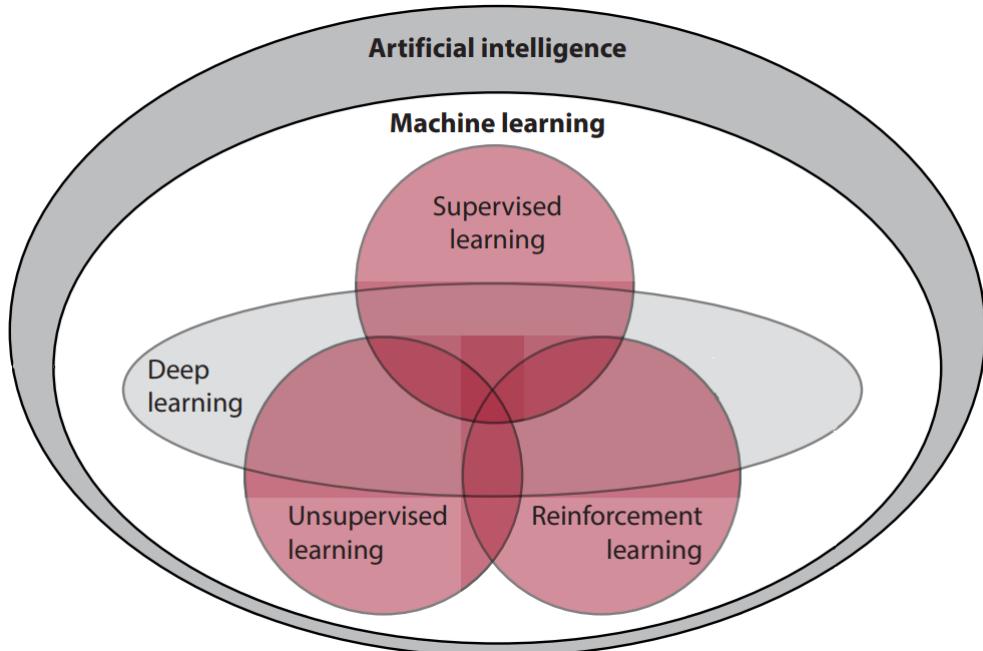


Figura 1.9: Principali ramificazioni del machine learning tratta da [13]

1.4.2 Deep Reinforcement Learning

Il *Deep Reinforcement Learning* (DRL) è semplicemente l'uso del Deep Learning per risolvere compiti di Reinforcement Learning.

Una definizione più accurata è fornita in [13]: il Deep Reinforcement Learning è un approccio di machine learning all'intelligenza artificiale che si occupa di sviluppare programmi informatici capaci di risolvere problemi che richiedono intelligenza. La caratteristica distintiva dei programmi di apprendimento profondo per rinforzo è quella di imparare attraverso trial-and-error, partendo da un feedback che è al tempo stesso sequenziale, valutativo e campionario, sfruttando potenti approssimazioni di funzioni non lineari.

La Figura 1.10 mostra il ciclo di interazione fondamentale nel processo di apprendimento del Reinforcement Learning. Questo può essere descritto in diverse fasi interconnesse. In primo luogo, l'agente osserva lo stato corrente dell'ambiente, un passaggio cruciale che fornisce le informazioni necessarie per decidere la prossima azione da intraprendere. Successivamente, l'agente sceglie un'azione da un insieme di possibilità, guidato dall'obiettivo di massimizzare le ricompense future. La selezione di questa azione è determinata da una policy, che può essere predefinita o sviluppata nel corso dell'apprendimento. L'azione scelta dall'agente provoca una transizione verso un nuovo stato ambientale, accompagnata da una ricompensa. Questa ricompensa può essere positiva, se l'azione è stata vantaggiosa, o negativa, se l'azione ha avuto esiti sfavorevoli.

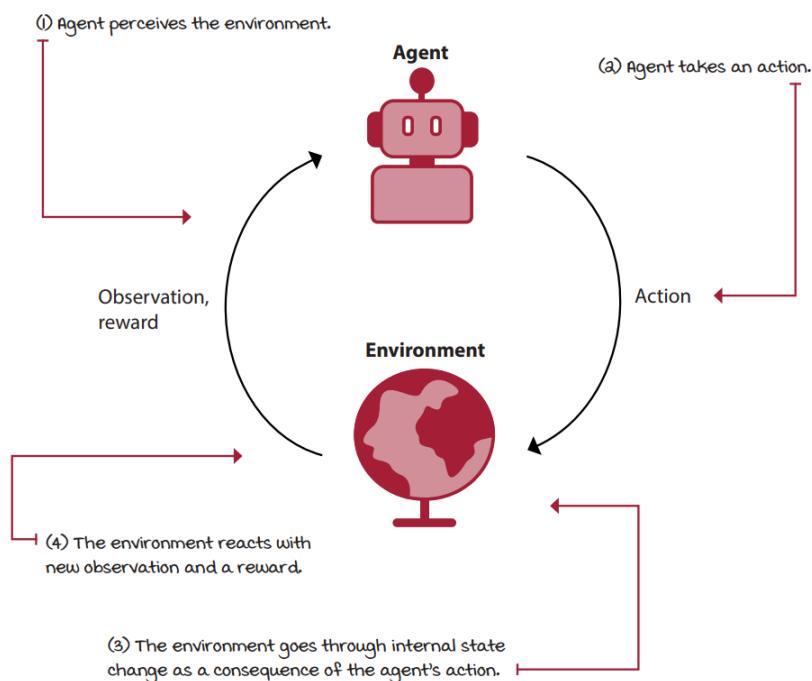


Figura 1.10: Illustrazione del ciclo di interazione nel Reinforcement Learning tratta da [13]

Successivamente, l’ambiente fornisce un feedback all’agente, presentandogli il nuovo stato e la ricompensa correlata. Questo feedback è essenziale per l’apprendimento dell’agente, il quale utilizza queste informazioni per aggiornare e perfezionare la propria policy di selezione delle azioni. Un episodio di apprendimento consiste in una serie di stati, azioni e ricompense che iniziano da uno stato iniziale e terminano in uno stato finale. Attraverso la sperimentazione e l’esperienza accumulate in ogni episodio, l’agente affina progressivamente la propria policy, migliorando le decisioni future.

1.4.3 Curriculum Learning

Spesso l’apprendimento per rinforzo viene applicato direttamente al problema da risolvere, svolgendo un processo di apprendimento specifico, di volta in volta. Viceversa, un obiettivo centrale del nostro lavoro era l’ottenimento di un modello generale, ottenuto da un processo di apprendimento anche elaborato, ma capace di portare allo sviluppo di competenze utili per risolvere problemi di movimento pedonale in una classe di ambienti abbastanza ampia.

Il *Transfer Learning* [22] rappresenta un miglioramento nell’apprendimento di un nuovo compito grazie al trasferimento di conoscenze da un compito correlato che è già stato appreso.

Negli ultimi anni, il Transfer Learning combinato con il Reinforcement Learning ha suscitato un notevole interesse nella comunità scientifica, rappresentando una direzione promettente per affrontare diverse sfide e risolvere problemi aperti nel campo del Reinforcement Learning. Studi approfonditi sul Transfer Learning applicato al Multi-Agent Reinforcement Learning sono stati condotti sia in [20] che in [21].

Il Transfer Learning viene suddiviso in due categorie principali: Intra-Agent Transfer e Inter-Agent Transfer. La prima categoria riguarda il riutilizzo della conoscenza appresa da un agente per risolvere nuovi compiti. La seconda categoria, invece, implica il trasferimento di conoscenza tramite la comunicazione tra agenti differenti.

All’interno della categoria dell’Intra-Agent Transfer, una metodologia rilevante è il *Curriculum Learning*. L’obiettivo del Curriculum Learning è migliorare sia la velocità di apprendimento sia le prestazioni finali degli agenti. Il curriculum consiste in una sequenza di compiti o ambienti con difficoltà incrementale in cui l’agente deve allenarsi. Gli agenti si allenano su un compito o in un ambiente fino a quando non acquisiscono i comportamenti necessari per superarlo, passando successivamente al compito seguente.

Il principio del Curriculum Learning è ispirato al modo naturale in cui esseri umani apprendono, ottenendo risultati migliori quando i concetti sono presentati in un ordine crescente di complessità. Secondo [5], il Curriculum Learning, nel contesto dell'apprendimento automatico, aiuta a migliorare notevolmente la generalizzazione dei modelli, accelerando la convergenza durante l'addestramento e migliorando la qualità dei minimi locali raggiunti.

1.4.4 PPO

Proximal Policy Optimization (PPO) [19] è un algoritmo di reinforcement learning basato su un'architettura di tipo *actor-critic*. Questa si compone di due reti neurali principali:

1. **Actor**: rete responsabile di selezionare le azioni da compiere data una certa osservazione dello stato dell'ambiente. L'attore apprende una politica, cioè una mappatura tra stati e azioni, che massimizza la ricompensa cumulativa attesa.
2. **Critic**: rete che valuta la qualità delle azioni suggerite dall'attore, stimando la funzione valore. In particolare, il critico calcola il *vantaggio* delle azioni eseguite, che misura quanto un'azione è migliore rispetto alla media delle azioni possibili in un dato stato.

Una delle principali innovazioni di PPO è l'introduzione di una funzione obiettivo surrogata che permette a un algoritmo on-policy di eseguire più step di gradiente sullo stesso mini-batch di esperienze. Questo approccio contrasta con i metodi on-policy tradizionali, come Advantage Actor Critic (A2C), che devono scartare immediatamente le esperienze dopo ogni passo di ottimizzazione.

La funzione obiettivo limitata ("clipped function", in inglese) di PPO permette di evitare che la politica si discosti troppo dopo ogni passo di ottimizzazione. Questa limitazione previene il collasso delle prestazioni dovuto all'elevata varianza intrinseca dei metodi di gradiente di politica on-policy, consentendo al contempo il riutilizzo dei mini-batch di esperienze per eseguire più passi di ottimizzazione.

PPO applica una strategia di limitazione simile anche agli aggiornamenti della funzione valore. Questa tecnica mantiene le variazioni dei Q-values entro limiti specifici, garantendo che le modifiche in termini di parametri siano regolari. Ciò è particolarmente utile per mantenere stabili le prestazioni del modello, anche quando le variazioni nello spazio dei parametri sono signifi-

cative.

Nel contesto della presente relazione di stage, si ritiene non necessario fornire una descrizione dettagliata degli algoritmi relativi alla funzione valore e alla funzione obiettivo. Questo approccio è giustificato dal fatto che il lavoro di stage non ha apportato modifiche sostanziali a tali algoritmi. Pertanto, si è scelto di concentrarsi sugli aspetti innovativi e sui risultati ottenuti durante il periodo di stage, riservando eventuali approfondimenti tecnici a fonti già esistenti e ben documentate in letteratura.

1.5 Valutazione modelli e validazione

La valutazione e l'eventuale validazione dei modelli ottenuti rappresentano il passo finale di ogni ciclo di esperimenti. Questo processo avviene attraverso test e metodologie condivise, permettendo il confronto con risultati esterni. Sebbene in campo di simulazione pedonale non esista uno standard universalmente riconosciuto, sono stati proposti schemi per test specifici, come quello descritto in [15], che fornisce una struttura per i test di evacuazione da edifici.

Struttura	Descrizione
Geometry	La configurazione del test
Scenario	Lo scenario di evacuazione da simulare
Expected result	Il risultato atteso che il modello di evacuazione deve produrre
Test method	Il metodo impiegato per il confronto tra il risultato atteso e i risultati simulati
User's actions	Le azioni richieste al tester durante l'esecuzione e la presentazione dei test

Tabella 1.1: Struttura proposta per i test in [15]

Nella maggior parte degli studi in questo ambito, vengono condotte analisi tramite grafici di misure considerate fondamentali. Tra queste, una delle più importanti è la relazione tra velocità media e la densità dei pedoni all'interno di uno spazio definito. Questo tipo di grafico fornisce numerose informazioni cruciali su come si comporta il flusso pedonale e identifica le quantità critiche che possono mettere in crisi il sistema.

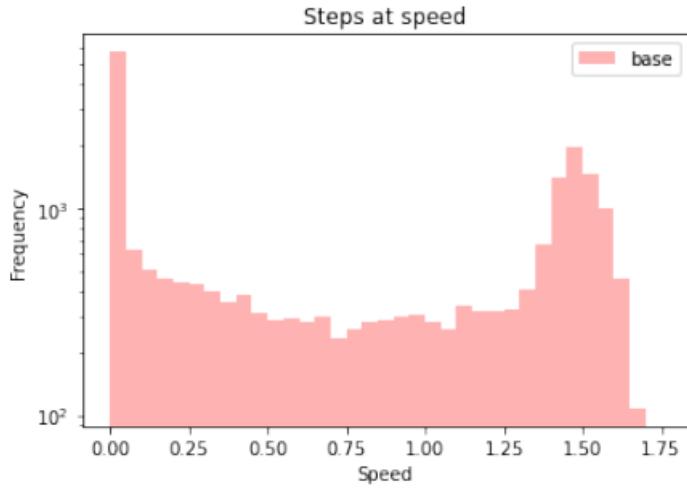


Figura 1.11: Istogramma di distribuzione della velocità di un pedone

La velocità può essere analizzata sotto diversi punti di vista, come il tempo e lo spazio. L'analisi temporale calcola per quanto tempo l'agente mantiene una determinata velocità, rivelando anche i periodi in cui i pedoni sono fermi e fornendo indicazioni utili per la modellazione. L'analisi spaziale mostra graficamente in quali sezioni dell'ambiente l'agente ha velocità maggiori, un ottimo strumento per individuare colli di bottiglia o sbilanciamenti.

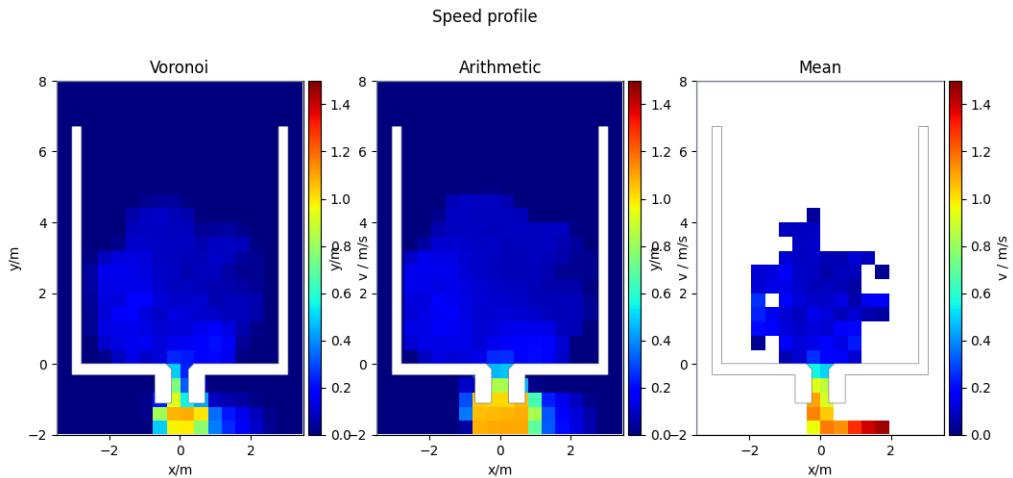


Figura 1.12: Esempio di profilo di velocità su un luogo da [18]

Capitolo 2

Un modello di DRL per la simulazione di pedoni

La prima parte della tesi si è concentrata sulla creazione di un sistema base in grado di sviluppare ambienti, monitorare i movimenti degli agenti e valutare i risultati conseguiti. Questo sistema è stato fortemente influenzato da ricerche precedentemente condotte, come descritto in [23], sebbene con l'adozione di strumenti e tecnologie differenti rispetto al modello attuale. Nella sezione successiva, verranno esaminati nel dettaglio tali strumenti e tecnologie impiegate per la realizzazione del sistema e per il conseguimento dei risultati descritti.

2.1 Strumenti

Gli strumenti e le metodologie principali utilizzati per condurre questo progetto sono descritti di seguito:

Godot Engine

Godot Engine¹ è stato scelto poiché permette di realizzare ambienti in maniera semplice, ma al tempo stesso fornisce una grande libertà nella personalizzazione e configurazione di tutti gli elementi al suo interno. Ad esempio, consente di gestire la fisica dei vari oggetti e realizzare script per il loro controllo utilizzando GDScript, un linguaggio di scripting integrato simile a Python.

Godot-RL-Agents

Godot-RL-Agents² è un pacchetto completamente open source che offre ai creatori di videogiochi e ai ricercatori nel campo dell'intelligenza artificiale, la possibilità di sviluppare comportamenti complessi per i loro personaggi non giocanti (NPC) o agenti. Questa libreria si integra perfettamente con Godot

¹<https://godotengine.org/>

²https://github.com/edbeeching/godot_rl_agents

Engine, permettendo di implementare facilmente algoritmi di apprendimento automatico.

Stable Baselines 3

Stable Baselines 3³ è una raccolta di implementazioni affidabili di algoritmi di apprendimento per rinforzo realizzati in PyTorch. Rappresenta la versione successiva e più avanzata di Stable Baselines, offrendo miglioramenti significativi in termini di prestazioni e facilità d'uso. È stato utilizzato per addestrare gli agenti nel nostro progetto, garantendo risultati stabili e replicabili.

PedPy

PedPy⁴ è una libreria Python open source, con licenza MIT, per l'analisi del movimento pedonale. Tale libreria è stata impiegata per analizzare i movimenti degli agenti, permettendo un'accurata valutazione delle loro dinamiche all'interno degli ambienti simulati.

2.2 Glossario

In questa sezione, si presenta un glossario dei termini rilevanti, utile per una comprensione approfondita dei concetti trattati nel capitolo.

L'**overfitting**, nel contesto del machine learning, si verifica quando un modello apprende troppo bene i dettagli e il rumore del set di dati di addestramento, al punto che le sue prestazioni su nuovi dati di test peggiorano. In altre parole, il modello diventa troppo complesso e specifico per i dati di addestramento, perdendo la capacità di generalizzare bene su dati non visti.

L'**underfitting**, nel contesto del machine learning, si verifica quando un modello è troppo semplice per catturare le tendenze e le relazioni sottostanti nei dati di addestramento. Di conseguenza, il modello ha scarse prestazioni sia sui dati di addestramento che sui dati di test, non riuscendo a rappresentare adeguatamente la complessità del problema.

³<https://github.com/DLR-RM/stable-baselines3>

⁴<https://github.com/PedestrianDynamics/PedPy>

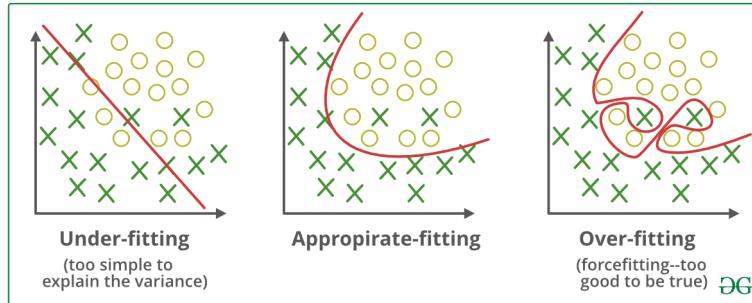


Figura 2.1: Rappresentazione di overfitting e underfitting

Gli **outliers**, ovvero i valori anomali, sono punti di dati che si discostano significativamente dagli altri punti del dataset. Gli outliers possono essere dovuti a vari motivi, come errori di misurazione, variazioni naturali o fenomeni imprevisti. Nel machine learning, i valori anomali possono influenzare negativamente l'addestramento dei modelli, distorcendo le stime e portando a prestazioni inferiori.

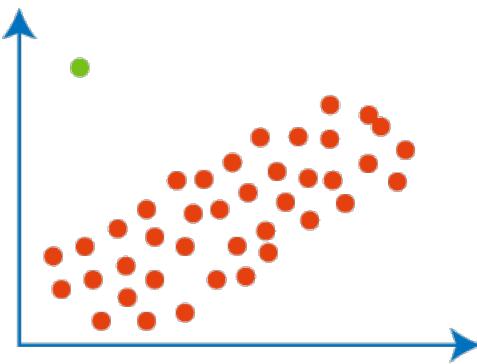


Figura 2.2: Esempio di valore anomalo

Gli **stalli in minimi locali** si verificano quando un algoritmo di ottimizzazione, come la discesa del gradiente, rimane bloccato in un minimo locale, ossia un punto in cui la funzione obiettivo ha un valore inferiore rispetto ai punti circostanti, ma non il valore minimo globale. Questo può impedire al modello di raggiungere le migliori prestazioni possibili, poiché l'algoritmo non riesce a superare questi minimi locali per trovare la soluzione ottimale globale.



Figura 2.3: Esempio di stallo in minimo locale

2.3 Ambiente

L’ambiente è stato creato con l’obiettivo di fornire un’area di simulazione per l’addestramento dell’agente. Lo spazio è considerato continuo e, pur essendo implementato in 3D, a livello modellistico è in sole due dimensioni, in quanto l’altezza non è mai rilevante; per questo viene visualizzato tramite una camera zenitale.

La Figura 2.4 mostra un esempio di ambiente contenente tutti gli elementi utilizzati per costruire gli ambienti del modello. Di seguito, viene fornita una descrizione degli elementi presenti nell’immagine:

1. **Pavimento:** si trova sotto l’intero ambiente e funge da base per il posizionamento degli altri elementi.
2. **Muro:** è invalicabile e l’agente non può vedere attraverso di esso. La forma e le dimensioni possono variare.
3. **Target finale:** rappresenta l’obiettivo ultimo dell’agente. Quando l’agente raggiunge il target finale, viene assegnato un grande reward. Anche la forma e le dimensioni del target finale possono variare.
4. **Agente:** si muove all’interno dell’ambiente delimitato dai muri, cercando di raggiungere il target finale. Altri eventuali agenti bloccano la visuale e sono invalicabili.
5. **Target intermedio:** è un obiettivo secondario dell’agente, spesso utilizzato per guidarne il percorso attraverso l’ambiente. La collisione con un target intermedio assegna un reward all’agente. Anche in questo caso, forma e dimensioni possono variare.

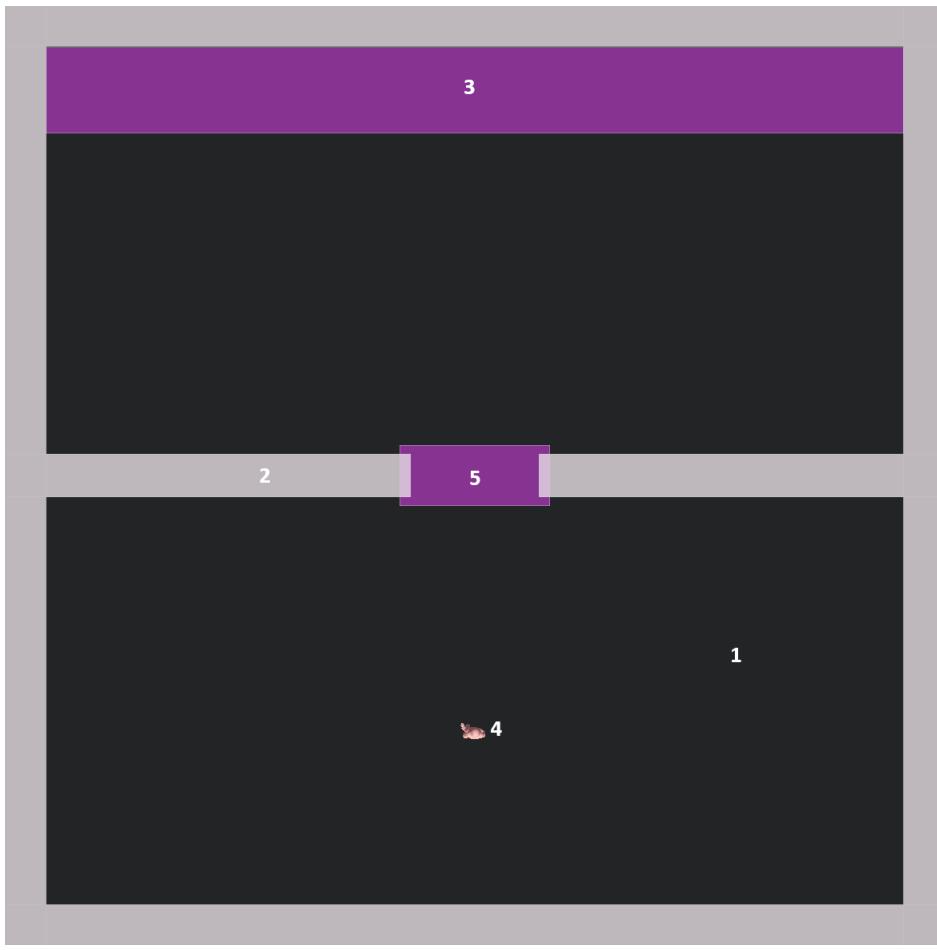


Figura 2.4: Esempio di un ambiente completo di dimensioni 20x20

2.4 Agente

Gli agenti utilizzati sono un'estensione degli agenti definiti nel framework Godot-RL-Agents, progettati specificatamente per apprendere attraverso la tecnica del reinforcement learning. Ogni agente, pur avendo dimensioni e capacità motorie identiche, è stato ulteriormente sviluppato per adattarsi meglio alle esigenze specifiche del nostro ambiente di simulazione. Le caratteristiche di base degli agenti sono le seguenti:

- Dimensioni: la forma del collisore del modello dell'agente è ottenuta tramite un collisore a capsula, con un'altezza di 1.8 metri e un raggio di 0.25 metri.
- Velocità e accelerazione massima: entrambe le caratteristiche verranno trattate in dettaglio nella Sezione 2.4.3.

2.4.1 Raggi

I sensori sono essenziali per un agente, in quanto permettono di raccogliere informazioni sull’ambiente circostante. Gli agenti implementati utilizzano due tipi di sensori: raggi, per la percezione della distanza, e archi, che aiutano a percepire la distanza sociale e personale con altre entità. Questi sensori simulano la capacità percettiva dell’agente e trasformano le informazioni raccolte in osservazioni utilizzabili dalla rete neurale per il processo di reinforcement learning. Questo permette all’agente di adattarsi e reagire in modo ottimale alle dinamiche dell’ambiente.

I raggi utilizzati dagli agenti sono creati mediante *raycast*. Questi si interrompono al contatto con un oggetto nell’ambiente, rilevando informazioni su distanza e tipo dell’oggetto impattato. Benché tutti i raggi forniscano queste informazioni di base, ne esistono vari tipi specifici per osservare differenti categorie di oggetti, come approfondiremo più avanti.

Per emulare la visione umana, i raggi non sono distribuiti uniformemente; sono infatti più densi al centro e si diradano progressivamente verso i lati. Questa particolare disposizione è regolata dall’Equazione 2.1, implementata per ottenere tale effetto di distribuzione.

$$\alpha_i = \min(\alpha_{i-1} + \delta \cdot i, \text{max_vision}) \quad (2.1)$$

Il parametro δ è fissato a 1.5, mentre il valore massimo di visione è impostato a 90 gradi e α_0 è inizializzato a 0. Di conseguenza, i raggi sono proiettati secondo le seguenti gradazioni: 0, 1.5, 4.5, 9, 15, 22.5, 31.5, 42, 54, 67.5, 82.5 e 90 gradi. Queste gradazioni sono distribuite sia in direzione positiva che negativa, garantendo agli agenti una visione complessiva di 180 gradi. Il numero totale di raggi restituiti dalla distribuzione è pari a 23, poiché il doppio raggio a 0 gradi viene eliminato in quanto superfluo.

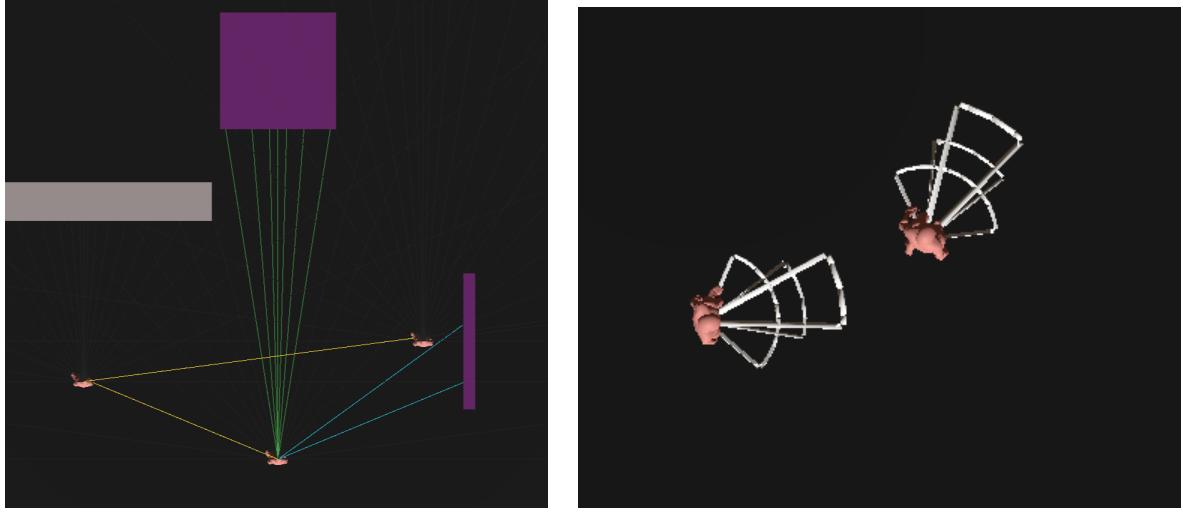
Tuttavia, è necessario disporre di diversi tipi di raggi in grado di fornire informazioni specifiche in base al tipo di oggetto osservato. Sono stati sviluppati due tipi principali di raggi:

- Raggi Muri+Target: Questi raggi sono progettati per rilevare collisioni con muri e target (sia i target intermedi che quelli finali).
- Raggi Muri+Agenti: Questi raggi sono configurati per rilevare collisioni con muri e altri agenti.

Di conseguenza, il numero totale di raggi per ciascun agente è fissato a 46. Una rappresentazione dei raggi è mostrato in Figura 2.5(a)

2.4.2 Archi

Al fine di fornire all’agente osservazioni riguardanti la prossemica, sono stati sviluppati degli archi con differenti raggio e ampiezza, progettati per rilevare la presenza di entità al loro interno. Questi archi possono rilevare sia muri che agenti. Gli archi sono posizionati a distanze di 0.6m, 1m e 1.4m. L’ampiezza di ciascun arco è determinata dal numero di raggi che contiene: l’arco meno ampio contiene 9 raggi, il secondo 11 raggi e l’arco più ampio ne contiene 17.



(a) I raggi di visione dell’agente

(b) Gli archi per la prossemica dell’agente

Figura 2.5: I raggi di visione e gli archi per la prossemica dell’agente

2.4.3 Spazio delle azioni

La rete neurale assegnata all’agente elabora le osservazioni raccolte dai sensori e le traduce in una serie di azioni di output. Queste azioni modificano la direzione e la velocità dell’agente secondo diverse regole. Ogni agente possiede una velocità desiderata media di 1.5 m/s con una deviazione standard di 0.2 m/s per simulare il passo di una persona. In questo modello, ogni agente può prendere tre decisioni al secondo, ognuna delle quali potrebbe cambiare la sua velocità e il suo orientamento.

Le azioni sono rappresentate da due valori continui nello spazio [-1,1] che vengono inseriti nelle equazioni utilizzate per modificare velocità e orientamento. Il primo valore, a_0 , viene utilizzato nell’Equazione 2.4 per modificare la velocità.

$$\text{speed}_t = \max \left(\text{speed}_{\min}, \min \left(\text{speed}_{t-1} + \frac{\text{speed}_{\max} \times a_0}{2}, \text{speed}_{\max} \right) \right) \quad (2.2)$$

Dove speed_{\min} è 0 e speed_{\max} è il valore assegnato all'agente secondo quanto detto in precedenza. L'equazione è modellata in modo che l'agente possa passare dalla velocità massima a fermarsi completamente in due azioni (circa 0.66 secondi).

Il secondo valore, a_1 , viene invece utilizzato nell'Equazione 2.3 per modificare la direzione dell'agente.

$$\alpha = \alpha + a_1 \times 25 \quad (2.3)$$

Modificando l'angolo in questo modo, l'agente può girarsi di un massimo di 25 gradi ad ogni azione (circa 0.33 secondi). L'obiettivo di questo modello è simulare la normale camminata di una persona in condizioni ordinarie.

2.4.4 Osservazioni

Le osservazioni sono normalizzate nell'intervallo $[0,1]$ per ottimizzare le prestazioni della rete neurale. L'agente raccoglie informazioni sia attraverso i raggi che in modo intrinseco. Nella Tabella 2.1 è riportato l'elenco delle diverse osservazioni.

Tipo	Osservazione	Valore
Intrinseco	Velocità	Numero
Raggio Muri + Target	Distanza	Numero
	Tipo	One Hot Encoding
Raggio Muri + Agenti	Distanza	Numero
	Tipo	Booleano
	Direzione	Numero
	Velocità	Numero

Tabella 2.1: Osservazioni

La prima osservazione rappresenta la velocità attuale dell'agente, normalizzata secondo la formula 2.4:

$$\text{Velocità} = \frac{\text{speed} - \text{speed}_{\min}}{\text{speed}_{\max} - \text{speed}_{\min}} \quad (2.4)$$

dove speed è la velocità attuale dell'agente, mentre speed_{\min} e speed_{\max} sono rispettivamente la velocità minima e la velocità massima. Questa osservazione permette all'agente di capire a che velocità si sta muovendo.

Successivamente, sono stati sviluppati due tipi di raggi: uno progettato per osservare esclusivamente i muri e i target, e un altro per monitorare i

muri e gli agenti. Nonostante possa inizialmente sembrare ridondante avere informazioni duplicate sui muri, è fondamentale dal momento che questi elementi sono barriere invalicabili. Escludendo i muri da uno dei due tipi di raggi, ciò consentirebbe la visualizzazione di agenti o target situati oltre i muri, compromettendo l'integrità dei dati osservati.

Si potrebbe ipotizzare un'alternativa unificazione dei raggi per ottenere un'unica osservazione di muri, target e agenti. Tuttavia, questa soluzione non consentirebbe di visualizzare contemporaneamente target e agenti, il che non rispecchierebbe fedelmente le dinamiche reali dell'ambiente studiato.

Muri e Target

I muri e i target sono osservati dall'agente tramite i raggi. I raggi catturano informazioni riguardo al tipo di entità osservata (muro o target) e alla loro distanza. Per normalizzare la distanza, il valore osservato viene diviso per 10 (lunghezza massima di un lato dell'ambiente), e tutti i valori maggiori vengono riportati come 1 (Equazione 2.5).

$$\text{Distanza} = \begin{cases} \frac{\text{dist}}{10} & \text{se } \text{dist} < 10 \\ 1 & \text{altrimenti} \end{cases} \quad (2.5)$$

Per riconoscere l'entità osservata è stato utilizzato un one hot encoding (Equazione 2.6).

$$\text{Tipo} = \begin{cases} [1, 0, 0] & \text{se entità = muro} \\ [0, 1, 0] & \text{se entità = mid target non visitato} \\ [0, 0, 1] & \text{se entità = mid target già visitato} \end{cases} \quad (2.6)$$

L'osservazione dei mid target è utile all'agente per capire se sta percorrendo la strada giusta o se sta tornando indietro, allontanandosi dal target finale. Poiché i raggi emessi sono 23 e ogni raggio, in grado di percepire muri e target, ritorna 4 osservazioni, in totale abbiamo 92 osservazioni.

Muri e Agenti

I muri e gli agenti presenti nell'ambiente sono percepiti dall'agente sempre tramite l'ausilio di raggi. La distanza è uguale ai raggi del tipo precedente (Equazione 2.7).

$$\text{Distanza} = \begin{cases} \frac{\text{dist}}{10} & \text{se dist} < 10 \\ 1 & \text{altrimenti} \end{cases} \quad (2.7)$$

Per quanto riguarda la distinzione dell'entità che si sta osservando, in questo caso basta utilizzare un booleano in quanto si deve distinguere tra due sole classi (Equazione 2.8).

$$\text{Tipo} = \begin{cases} 1 & \text{se entità} = \text{agente} \\ 0 & \text{se entità} = \text{muro} \end{cases} \quad (2.8)$$

La terza osservazione riguarda la direzione relativa all'agente osservato. Essa viene calcolata come il prodotto scalare tra la propria direzione e la direzione dell'agente osservato, convertendola poi in un range [0,1] (Equazione 2.9).

$$\text{Direzione} = \begin{cases} \frac{(\text{dir}_{\text{agente}} \cdot \text{dir}_{\text{agente_oss}}) + 1}{2} & \text{se entità} = \text{agente} \\ 0 & \text{se entità} = \text{muro} \end{cases} \quad (2.9)$$

L'ultima osservazione è atta a misurare la velocità dell'agente che si sta osservando, normalizzata con la formula di normalizzazione della velocità vista precedentemente (Equazione 2.10).

$$\text{Velocità} = \begin{cases} \frac{\text{speed} - \text{speed}_{\min}}{\text{speed}_{\max} - \text{speed}_{\min}} & \text{se entità} = \text{agente} \\ 0 & \text{se entità} = \text{muro} \end{cases} \quad (2.10)$$

Nel caso in cui si stia osservando un muro, la direzione e la velocità saranno pari a 0. Anche in questo caso i raggi emessi sono 23 e da ognuno di essi vengono estratte 4 osservazioni, portando il numero totale di osservazioni, per questo tipo di raggi, a 92.

Osservazioni Totali

In totale, abbiamo una misura intrinseca, 92 osservazioni provenienti dal primo tipo di raggi e 92 osservazioni dal secondo tipo di raggi, portando il numero totale di osservazioni a 185.

$$n_{\text{oss}} = 185 = 1 + 23 \cdot 4 + 23 \cdot 4 \quad (2.11)$$

2.5 Rewards

I rewards sono il meccanismo alla base del reinforcement learning in quanto vanno a premiare o punire le azioni e interazioni dell'agente. A seconda di come vengono definiti i rewards l'agente svilupperà vari comportamenti diversi e avrà preferenze e bias verso alcuni pattern di movimento; uno studio sperimentale approfondito sui valori di reward è quindi fondamentale per la definizione di un agente efficace. Di seguito vengono elencati i rewards utilizzati per il modello base.

$$\text{Reward} = \begin{cases} +6 & \text{Raggiunto target finale} \\ +0.5 & \text{Raggiunto target intermedio non ancora visitato} \\ -1 & \text{Raggiunto target intermedio già visitato} \\ -0.5 & \text{Nessun target in vista} \\ -0.5 & \text{Collisione con un muro} < 0.6\text{m} \\ -0.5 & \text{Collisione con un agente} < 0.6\text{m} \\ -0.005 & \text{Vicinanza con un agente} < 1.0\text{m} \\ -0.001 & \text{Vicinanza con un agente} < 1.4\text{m} \\ -0.0001 & \text{Trascorso un timestep} \\ -6 & \text{Scaduti timesteps massimi} \end{cases}$$

Dettagli rewards

In questa sezione vengono spiegate nel dettaglio i vari rewards, analizzando il motivo della scelta dei valori e della struttura per ottenere il comportamento desiderato.

Raggiunto target finale (+6) Quando l'agente raggiunge il target finale, viene assegnata la ricompensa più alta. Questa ricompensa è fondamentale poiché insegna all'agente che, nonostante possa incontrare ricompense negative durante il suo percorso, il raggiungimento dell'obiettivo finale gli garantirà una grande ricompensa, compensando ampiamente le penalità precedenti.

Raggiunto target intermedio non ancora visitato (+0.5) Quando l'agente raggiunge un target intermedio non ancora visitato, viene fornita una ricompensa positiva poiché tali target sono posizionati per guidare l'agente nei percorsi ottimali per risolvere l'ambiente.

Raggiunto target intermedio già visitato (-1) Se l’agente raggiunge un target intermedio già visitato, riceve una penalità in quanto non ha seguito il percorso corretto per raggiungere il target finale.

Nessuna osservazione di target (-0.5) Se i raggi dell’agente non rilevano nessun target intermedio, viene assegnata una penalità, poiché questo comportamento indica che l’agente non sa dove si trovi il suo obiettivo (intermedio o finale) e avrà difficoltà a completare l’ambiente.

Collisione con un muro (entro 0.6m) (-0.5) Se l’agente rileva una collisione con un muro, viene assegnata una penalità. Questo comportamento è scoraggiato poiché non è realistico e può causare problemi agli agenti, che potrebbero rimanere bloccati vicino ai muri e avere difficoltà a riprendere il percorso corretto.

Collisione con un altro agente (entro 0.6m) (-0.5) Se l’agente rileva una collisione con un altro agente, viene assegnata una penalità. Questo comportamento è altamente scoraggiato poiché può creare situazioni di stallo con continue collisioni.

Vicinanza con un altro agente (entro 1m) (-0.005) Se l’agente rileva la presenza di un altro agente a una distanza inferiore a 1 metro, viene assegnata una piccola penalità. Questo comportamento è moderatamente scoraggiato poiché potrebbe portare l’agente a scontrarsi con un altro agente se non corregge la propria traiettoria. Questa penalità funge da avvertimento per l’agente a cambiare direzione.

Vicinanza con un altro agente (entro 1.4m) (-0.001) Se l’agente rileva la presenza di un altro agente a una distanza inferiore a 1.4 metri, viene assegnata una penalità lieve. Questo comportamento serve rilevare la presenza di un altro agente nelle vicinanze, dandogli tempo e spazio per cambiare direzione ed evitare una collisione.

Trascorso un timestep (-0.0001) Ogni timestep trascorso comporta una piccola penalità. Questa penalità è importante per insegnare all’agente a preferire una soluzione veloce rispetto a una lenta. La regolazione di questo parametro può influenzare drasticamente il comportamento dell’agente, imponen-

do maggiore o minore urgenza e potenzialmente portando a comportamenti indesiderati se l'urgenza è eccessiva.

Scaduti i timesteps massimi (-6) Se l'agente non riesce a completare l'ambiente entro il numero massimo di timesteps previsti, l'episodio di training fallisce, e viene assegnata la massima penalità negativa (pari in valore assoluto alla ricompensa per il successo). Questa penalità, insieme a quella per ogni timestep trascorso, insegna all'agente a risolvere gli ambienti entro il tempo previsto.

2.6 Curriculum

Il curriculum del modello base è composto da 11 ambienti, mostrati nella Tabella 4.1 e spiegati brevemente per caratteristiche e scopo.

Comportamento	Ambiente	Retraining
Camminare verso il target	StartEz	No
	Start	Sì
Guardare il target	Osserva	Sì
Navigare in corridoi stretti	StartCorridoio	No
Curvare e evitare prossemica muri	Curve	No
	CurveOstacoli	Sì
Evitare agenti nella stessa direzione	PortaStretta	Sì
Evitare agenti in direzioni diverse	Corridoio3vs3	Sì
	Incroci4	Sì
	Giuunzione a T	Sì
Combinazione di tutti i comportamenti	DoppioPortaStretta	Sì

Tabella 2.2: Ambienti d'addestramento

StartEz

Ambiente iniziale, l'agente è posizionato vicino al target finale, come mostrato in Fig 2.6(a). Il fine primario è quello di introdurre l'agente alle interazioni dirette con i target finali, evidenziando come queste conducano a ricompense significative. La posizione dell'agente e del target finale sono fissati.

Start

In questo secondo ambiente, l'agente e l'obiettivo finale vengono posizionati casualmente all'inizio di ogni episodio. L'obiettivo è far sì che l'agente sviluppi la capacità di cercare l'obiettivo finale anche quando non è direttamente visibile, potenziando così la sua capacità di navigazione e di ricerca.

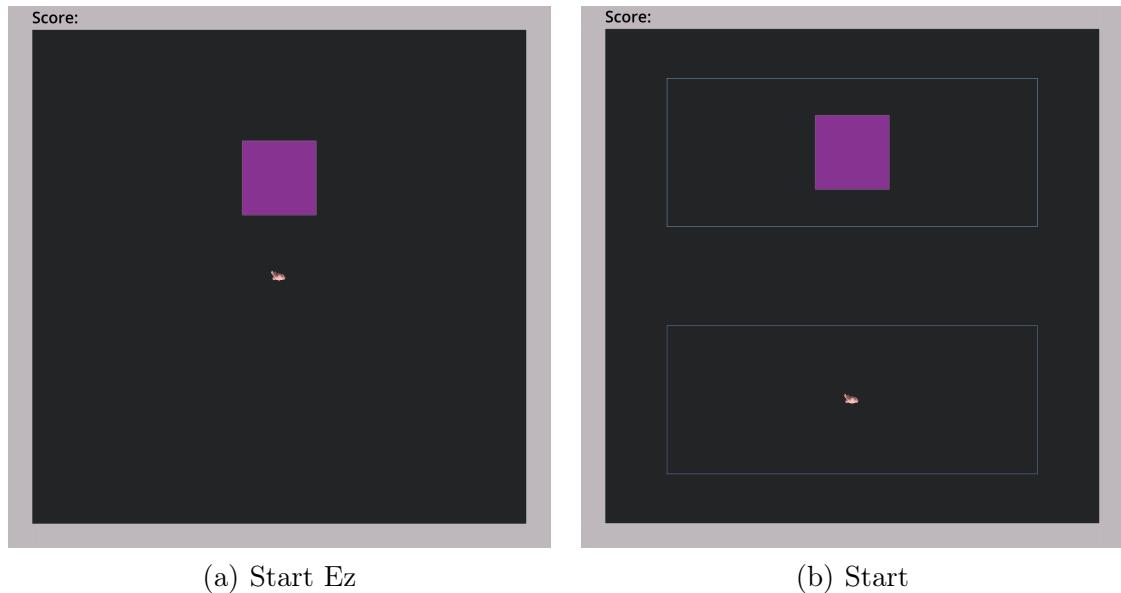


Figura 2.6: Ambienti “StartEz” e “Start” del curriculum.

Osserva

In questo ambiente, l'agente è limitato nella mobilità, potendo solo ruotare su se stesso per esaminare gli elementi circostanti. L'ambiente è caratterizzato da ostacoli e da un obiettivo finale posizionato casualmente all'inizio di ogni episodio. Senza la capacità di spostarsi fisicamente, per l'agente diventa impossibile raggiungere l'obiettivo finale e conseguire un punteggio positivo. Tuttavia, l'agente ha la capacità di mantenere l'obiettivo finale costantemente in vista, cercando così di minimizzare le penalità. L'obiettivo principale di questo ambiente è quindi sviluppare nell'agente abilità di osservazione avanzate.

Start Corridoio

In questo ambiente, l'agente è situato al centro di un corridoio, con l'obiettivo finale che appare casualmente a una delle due estremità. La finalità principale di questo scenario è di abituare l'agente alle interazioni con i muri, dato che all'avvio dell'episodio si trova in prossimità immediata di uno di essi, migliorare le sue strategie di localizzazione degli obiettivi, che non sono visibili all'inizio, e insegnargli a muoversi in linea retta.

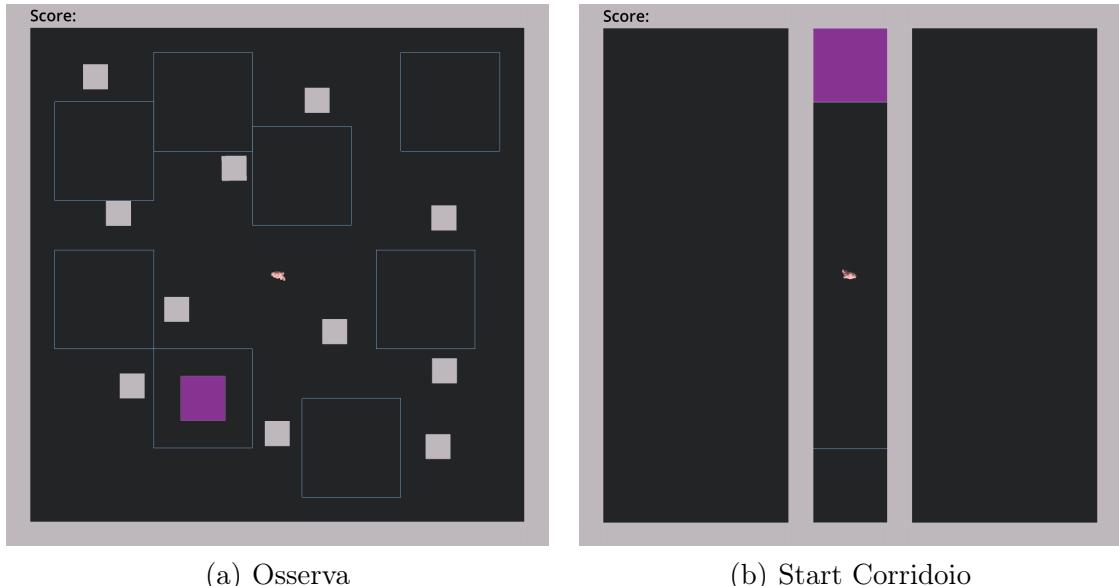


Figura 2.7: Ambienti “Osserva” e “Start Corridoio” del curriculum.

Curve

L’ambiente Curve consiste in un corridoio estremamente lungo, caratterizzato da cinque curve strette e angolate che l’agente deve affrontare prima di raggiungere l’obiettivo finale. L’obiettivo principale di questo ambiente è far acquisire all’agente familiarità con le curve in corridoi molto stretti. Per evitare il fenomeno dell’adattamento eccessivo e migliorare le capacità di generalizzazione dell’agente, una volta completato un episodio, l’ambiente ha una probabilità del 50% di essere ruotato di 90 gradi.

Curve Ostacoli

Questo ambiente presenta una progressione rispetto al precedente mantenendo la struttura di un unico corridoio lungo. Per aumentare la complessità, nei tratti rettilinei del corridoio, le pareti presentano sporgenze che costituiscono ostacoli aggiuntivi. Così come l’ambiente Curve, anche Curve Ostacoli vien ruotato per favorire la generalizzazione ed evitare adattamento eccessivo.

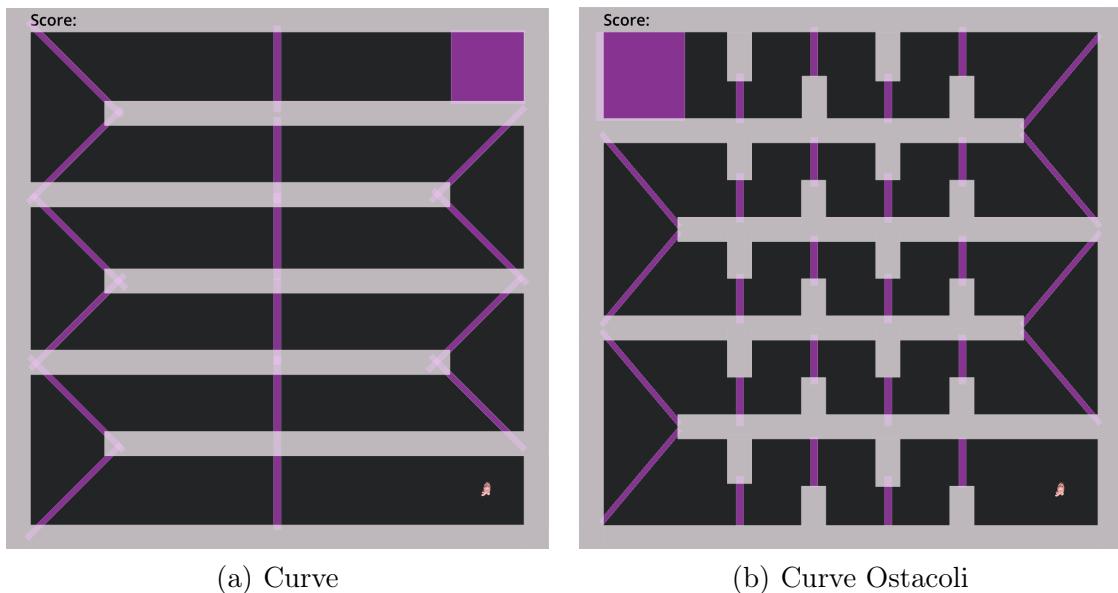


Figura 2.8: Ambienti “Curve” e “Curve Ostacoli” del curriculum.

Porta Stretta

In questo ambiente, si introduce per la prima volta la presenza di più agenti. Gli agenti vengono posizionati all'inizio di un ampio spazio prima di una piccola entrata che contiene un target intermedio. Per prevenire congestioni alla porta o subire penalità rilevanti, è necessario che gli agenti si coordinino e collaborino efficacemente per superare con successo le sfide poste dall'ambiente. Il passaggio viene posizionato casualmente all'interno di un'area specifica intorno al centro dell'ambiente all'inizio di ogni episodio.

Corridoio 3vs3

Questo ambiente rappresenta un'evoluzione rispetto all'ambiente Start Corridoio. In questo caso, sono presenti due gruppi opposti di agenti, ognuno con il compito di percorrere lo stesso corridoio. L'obiettivo cruciale qui è evitare collisioni e ingorghi, permettendo agli agenti di superare l'ambiente in modo efficiente.

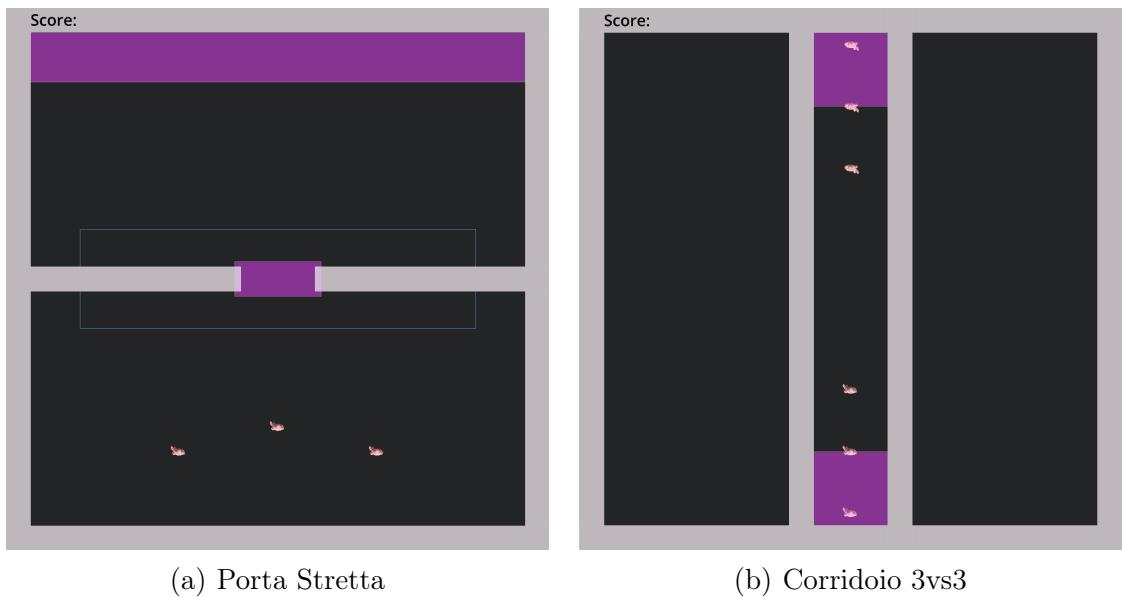


Figura 2.9: Ambienti “Porta Stretta” e “Corridoio 3vs3” del curriculum.

Incrocio4

Questo ambiente presenta un incrocio, con un gruppo di quattro agenti posizionato nella parte sinistra e un altro gruppo di quattro agenti posizionato nella parte inferiore. L'obiettivo di ciascun gruppo è raggiungere il lato opposto dell'ambiente. Tuttavia, per farlo, gli agenti devono affrontare un incrocio complesso formato da otto agenti in totale. Questa configurazione richiede un'interazione e una coordinazione avanzate tra i gruppi di agenti per raggiungere i rispettivi obiettivi senza collisioni o intoppi.

Giunzione a T

In questo ambiente, è presente un incrocio in cui si congiungono due flussi provenienti da due direzioni opposte. Al centro dell'incrocio, i due flussi si uniscono per raggiungere il target finale posizionato in alto. Target intermedi assistono nell'orientamento e nella percorrenza del corridoio. L'obiettivo principale di questo ambiente è insegnare agli agenti a mantenere una traiettoria minimale nelle curve, evitando di intralciare agenti appartenenti ad altri flussi.

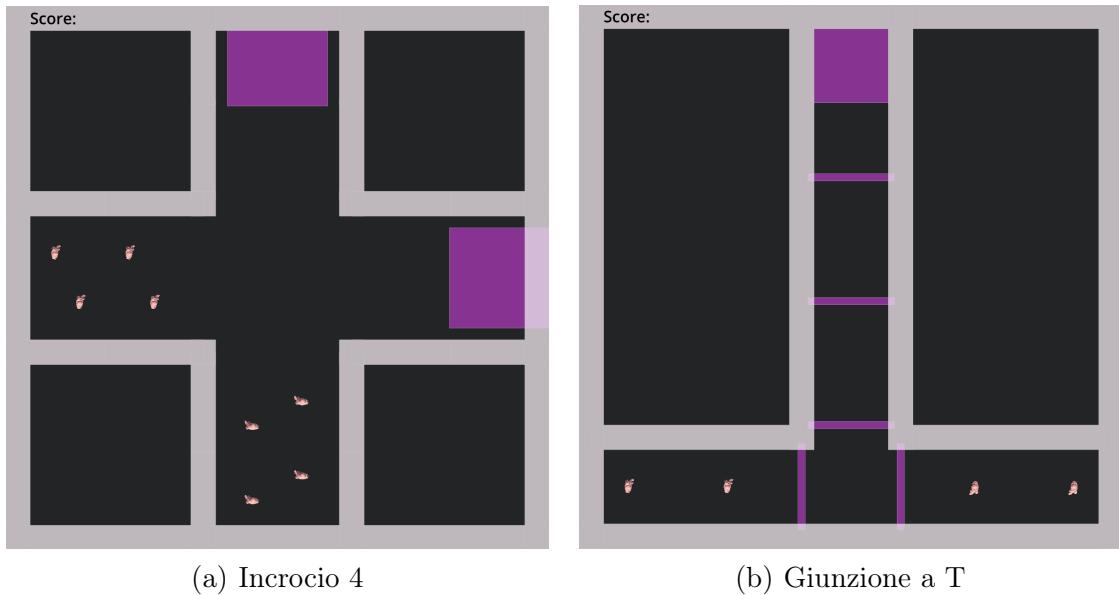
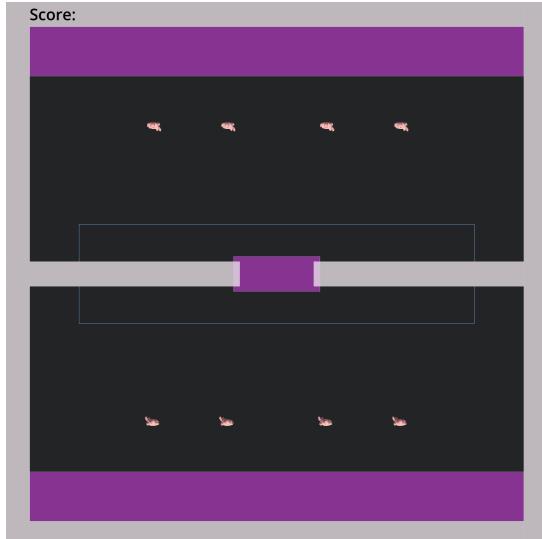


Figura 2.10: Ambienti “Incrocio 4” e “Giunzione a T” del curriculum.

Doppio Porta Stretta

L'ultimo ambiente rappresenta un'evoluzione di Porta Stretta e introduce un possibile ingorgo che divide due flussi, ognuno composto da quattro agenti, diretti verso i rispettivi target finali. L'obiettivo principale di questo

ambiente è potenziare la capacità di coordinazione tra agenti indipendenti nell’attraversamento di spazi comuni con membri del proprio flusso o con flussi antagonisti.



(a) Doppio Porta Stretta

Figura 2.11: Ambiente “Doppio Porta Stretta” del curriculum.

2.7 Addestramento

L’addestramento dell’agente avviene attraverso la somministrazione graduale di ambienti sempre più difficili, come descritto nel dettaglio nella sezione precedente. Il passaggio da un ambiente all’altro è regolato da un punteggio medio minimo necessario: se questo valore viene superato, il training sull’ambiente corrente termina e si attiva l’ambiente successivo.

Il punteggio minimo viene calcolato ogni volta che vengono completati 36 episodi. Al termine di ciascun episodio (quando vengono raggiunti i time-steps massimi o tutti gli agenti hanno raggiunto il target finale), la media del punteggio ottenuto dagli agenti in quell’episodio viene salvata in una struttura dati. Una volta completati i 36 episodi, viene calcolata la media sfrondata, escludendo il 10% dei valori più alti e più bassi per evitare che valori anomali positivi o negativi influenzino il punteggio medio. Il valore ottenuto viene quindi confrontato con lo score medio minimo richiesto: se è superiore, l’agente passa al prossimo ambiente; altrimenti, inizia un altro ciclo di 36 episodi.

2.7.1 Early Fail

L’addestramento tramite reinforcement learning del caso di studio corrente non garantisce la convergenza teorica, considerata la complessità del task e la possibilità di incorrere in esecuzioni particolarmente ostiche durante l’addestramento. Per affrontare questo problema, è stato implementato un sistema di *early fail*.

Il funzionamento del sistema di *early fail* prevede un numero massimo di episodi associato a ciascun ambiente. Una volta superato questo limite, l’addestramento viene interrotto completamente e la run di addestramento viene considerata un fallimento. Prese in considerazione la media e la varianza degli ambienti, sono state create 3 classi di ambienti per determinare il numero massimo di cicli. Considerando che si utilizzano 36 episodi per calcolare la media e verificare se questa supera la soglia minima, abbiamo definito un numero massimo di cicli per poter eseguire una nuova run, in caso non si superi tale soglia:

- Negli ambienti più semplici, sono stati concessi 5 cicli, per un totale di 180 episodi.
- Negli ambienti di media difficoltà, sono stati forniti 10 cicli, per un totale di 360 episodi.
- Negli ambienti più complessi, sono stati concessi 15 cicli, per un totale di 540 episodi.

2.7.2 Fasi di Addestramento

L’addestramento dell’agente è composto da due fasi: training e retraining.

Training

Il training è caratterizzato dalla somministrazione di ambienti di difficoltà sempre maggiore, in accordo con i principi del *curriculum learning*. Per la fase di training si utilizzano 10 copie dello stesso ambiente in parallelo per velocizzare il processo. Una volta che lo score minimo dell’ambiente è raggiunto, viene somministrato l’ambiente successivo del curriculum. Alla fine dell’ultimo ambiente di training inizia la fase di retraining.

Retraining

Il retraining consiste nella ripetizione degli ambienti del curriculum di training con un numero di episodi massimi limitato. Vengono utilizzate 3 istanze

per ogni ambiente, e l’addestramento termina dopo un numero di timesteps massimi prestabilito. Questa fase serve per riprendere alcuni degli insegnamenti che potrebbero perdersi durante il processo di training, specialmente se si protrae molto nel tempo.

2.7.3 Risultati addestramento

Risorse hardware utilizzate

Il modello base è stato addestrato più volte su macchine diverse per verificarne la consistenza e stimarne la variabilità. Le due macchine utilizzate supportano Windows 10 e Windows 11 come sistema operativo e hanno la seguente configurazione hardware:

Macchina	CPU	GPU	RAM
Macchina 1	Intel i7-10700F 2.90 GHz	NVIDIA GeForce RTX 3060 12GB	16 GB 3600 MHz
Macchina 2	AMD Ryzen 7 3700X 3.59 GHz	NVIDIA GeForce RTX 3070Ti 8GB	16 GB 3600 MHz

Tabella 2.3: Specifiche delle macchine utilizzate

L’addestramento è stato eseguito molteplici volte per testare la consistenza dell’utilizzo di risorse. Per quanto riguarda memoria e CPU utilizzata, i valori risultano costanti per tutte le esecuzioni e sono riportati in Tabella 2.4:

Macchina	CPU Utilizzata		RAM Utilizzata	
	Python	Godot	Python	Godot
Macchina 1				
Singolo Agente	22%	17%	243MB	235MB
Multi Agente	55%	36%	243MB	241MB
Retraining	68%	44%	243MB	255MB
Macchina 2				
Singolo Agente	21%	13%	255MB	238MB
Multi Agente	58%	33%	255MB	239MB
Retraining	60%	36%	255MB	246MB

Tabella 2.4: Risorse utilizzate in fase di training

Andamento reward

L’analisi del grafico della reward è fondamentale per valutare l’efficacia del curriculum learning. Questo grafico mostra l’andamento della reward media durante l’addestramento, con il cambio degli ambienti nel curriculum.

La parte più interessante è quella relativa agli ambienti di training, mentre la parte dedicata al retraining rappresenta una media di tutti gli ambienti somministrati in parallelo.

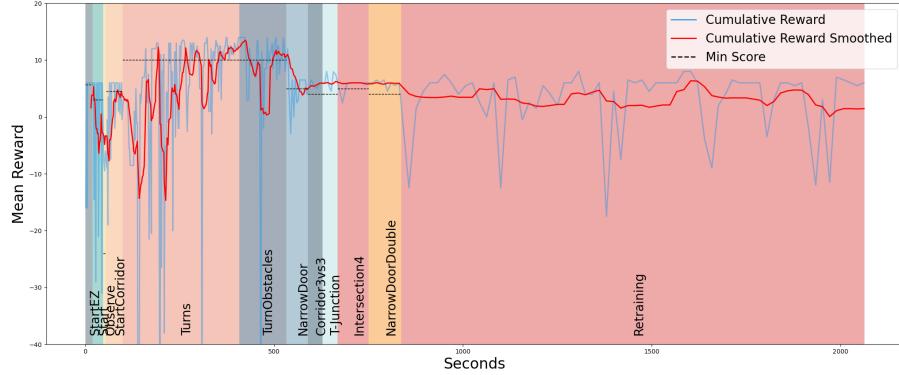


Figura 2.12: Reward Cumulativo

In generale, si osserva un aumento della reward nel tempo, con alcune cadute coincidenti con la fine dell'addestramento degli ambienti che raggiungono il punteggio minimo richiesto. Questo è dovuto al fatto che l'agente si concentra sugli ambienti che ancora necessitano di miglioramento.

Alcuni ambienti, come “Start Corridoio” e “Curve Ostacoli”, mostrano un andamento della reward in costante aumento. Altri, invece, come “Curve”, presentano fasi di discesa alternate a fasi di salita. Questo comportamento è dovuto alla difficoltà di questi ambienti e alla necessità per l'agente di imparare nuovi comportamenti rispetto a quelli necessari per gli ambienti precedenti.

Per visualizzare invece la durata dei singoli episodi e il tempo totale dell'esecuzione possiamo utilizzare la Figura 2.13

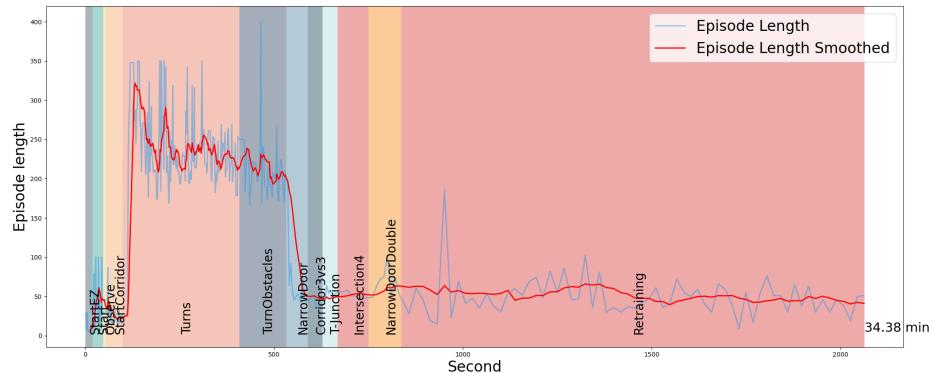


Figura 2.13: Lunghezza degli episodi

2.8 Test

La fase di test serve a misurare quanto efficacemente gli agenti, addestrati secondo il metodo presentato precedentemente, possono applicare le competenze acquisite in contesti nuovi e non previsti.

2.8.1 Ambienti di Test

Di seguito sono riportati gli ambienti di test. È importante notare che questi ambienti non sono noti agli agenti, quindi possono essere considerati una valutazione autentica della loro capacità di generalizzazione.

Curva C

L'ambiente Curva C, illustrato nella Figura 2.14(a), è estremamente basilare, caratterizzato dalla presenza di un unico agente che deve eseguire due curve semplici. Nonostante la sua semplicità, questo ambiente può rivelarsi utile per identificare comportamenti anomali.

Omega

L'ambiente Omega, mostrato nella Figura 2.14(b), è un ambiente di test che coinvolge un singolo agente incaricato di navigare attraverso diverse curve. Questo scenario è cruciale per valutare l'efficacia del modello nella navigazione.

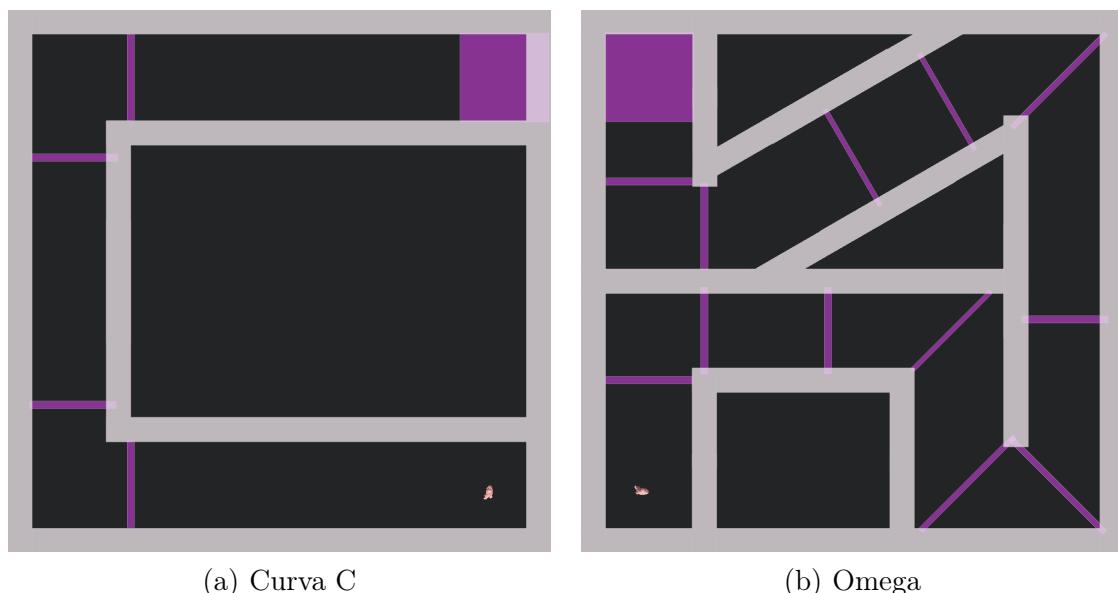


Figura 2.14: Ambienti “Curva C” e “Omega” della fase di test

Ancora

L'ambiente di test Ancora, illustrato nella figura 4.8(a) coinvolge quattro agenti divisi in due gruppi, che partono da corridoi opposti. Dopo un percorso rettilineo, si incontrano dietro una curva con un angolo cieco e devono coordinarsi per attraversare l'ultimo corridoio verso l'obiettivo finale, testando così la loro capacità di evitare congestioni.

Scelta Porta

L'ambiente Scelta Porta, illustrato nella figura 2.15(b), presenta un agente in una posizione casuale nella parte inferiore e l'obiettivo nella parte superiore. Tra loro ci sono due muri: il primo con due porte, il secondo con una sola porta. Questo ambiente testa la capacità decisionale degli agenti quando devono scegliere tra due alternative, elemento non presente negli ambienti di addestramento.

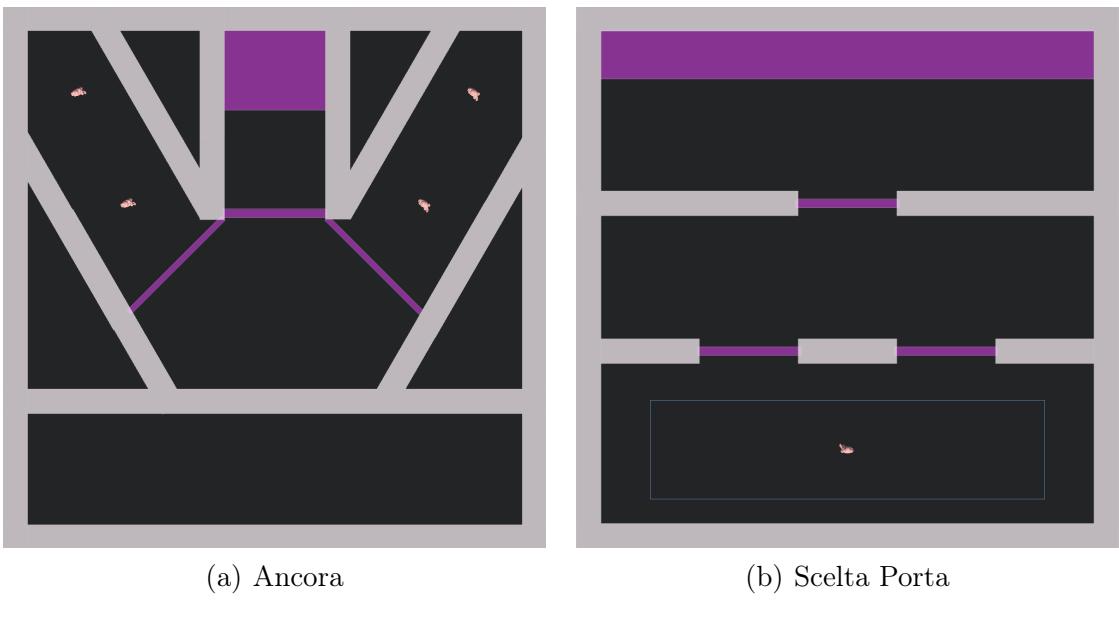


Figura 2.15: Ambienti “Ancora” e ”Scelta Porta” della fase di test

Doppia Porta Opposta

L'ambiente di test Doppia Porta Opposta, presentato nella Figura 2.16(a), include sei agenti divisi in due gruppi di tre. Un corridoio centrale, accessibile attraverso due porte, separa i due gruppi. Gli agenti sono chiamati a coordinarsi con precisione per attraversare simultaneamente le porte e il corridoio centrale, evitando così eventuali congestioni. La complessità di

questo ambiente è accentuata sia dalla presenza di numerosi agenti sia dalla necessità di un'attenta coordinazione a causa delle doppie porte.



(a) Doppia Porta Opposta

Figura 2.16: Ambiente “Doppia Porta Opposta” della fase di test

2.8.2 Analisi dei risultati

In questa sezione vengono esaminati i risultati ottenuti dai dati raccolti sugli agenti all'interno dell'ambiente di simulazione. Per questa analisi vengono eseguite 10 ripetizioni per ciascun ambiente. Ogni secondo vengono registrati 30 campionamenti riguardanti gli agenti. Quando gli agenti raggiungono l'obiettivo finale un numero prestabilito di volte, i dati vengono salvati su file per effettuare l'analisi.

I dati raccolti vengono successivamente elaborati in un notebook Python. Utilizzando PedPy, esaminiamo il comportamento degli agenti negli ambienti di prova, consentendo una comprensione dettagliata e approfondita del loro comportamento nelle diverse situazioni simulate.

Per il seguente modello, la visualizzazione più utile al nostro scopo, è quella delle traiettorie, in quanto ci permette di visualizzare come gli agenti affrontano i nuovi livelli e siano capaci di portare a termine l'ambiente, raggiungendo l'uscita delle stanze.

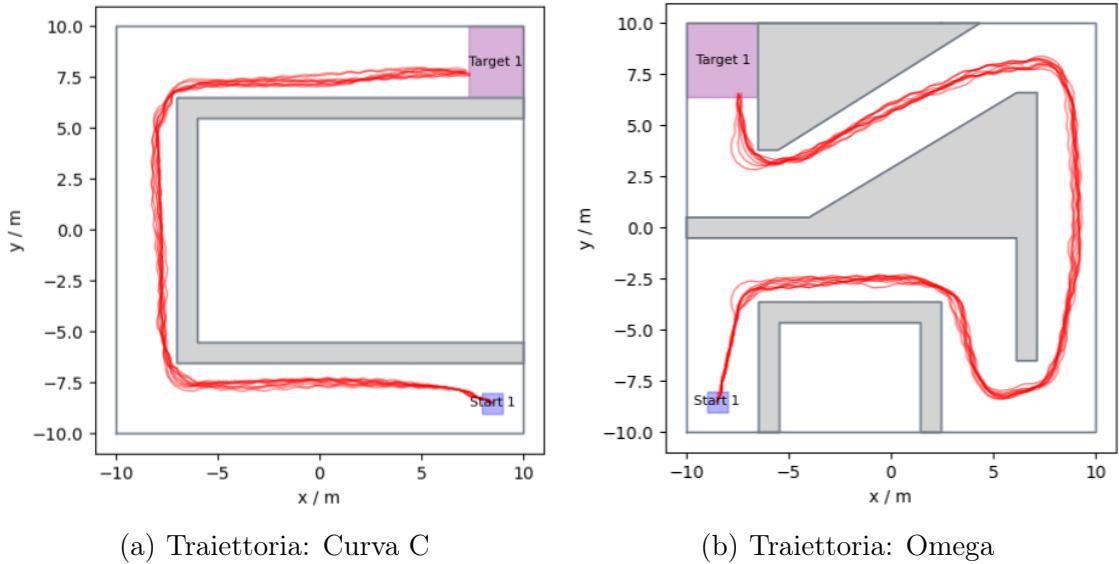
Successivamente, nel modello esteso, andremo ad aumentare il traffico all'interno degli ambienti, permettendo di andare ad analizzare anche le densità, le velocità, tempi e i comportamenti di coordinamento tra pedoni dello stesso gruppo e di gruppi diversi.

Curva C

Nell’ambiente “Curva C”, le traiettorie degli agenti seguono con precisione le due curve previste, dimostrando un comportamento coerente e prevedibile. Gli agenti completano il percorso senza deviazioni significative, mostrando la loro capacità di mantenere la traiettoria anche in presenza di curve.

Omega

Nell’ambiente “Omega”, le traiettorie degli agenti mostrano una navigazione fluida e senza interruzioni attraverso le curve. Gli agenti gestiscono con successo la varietà di curve presenti nel percorso, dimostrando un’efficace capacità di navigazione.



(a) Traiettoria: Curva C

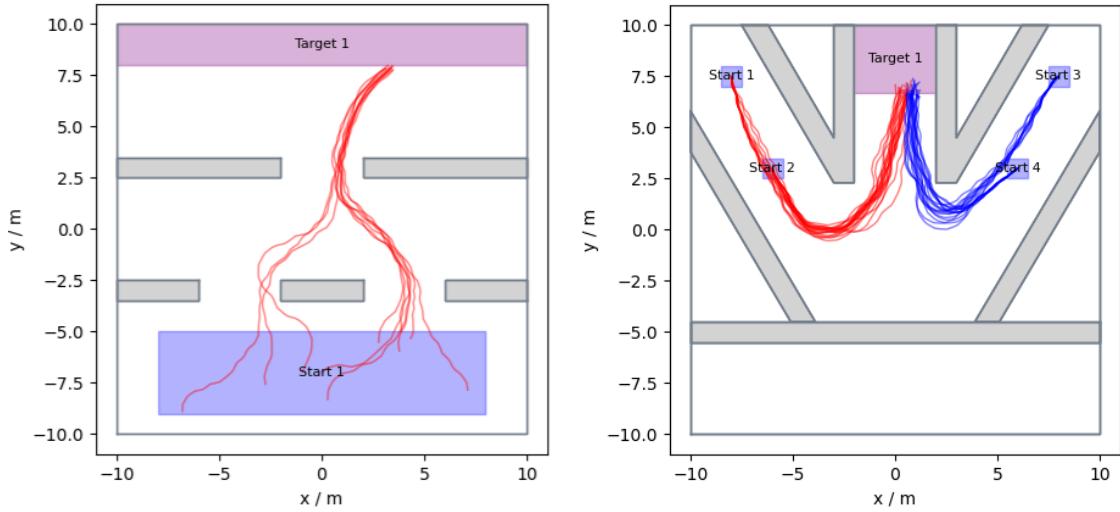
(b) Traiettoria: Omega

Scelta Porta

Le traiettorie nell’ambiente “Scelta Porta” dimostrano la capacità decisionale degli agenti. Gli agenti scelgono la porta più vicina al punto di partenza e proseguono attraverso la porta del secondo muro senza problemi, evidenziando una navigazione efficace e priva di difficoltà.

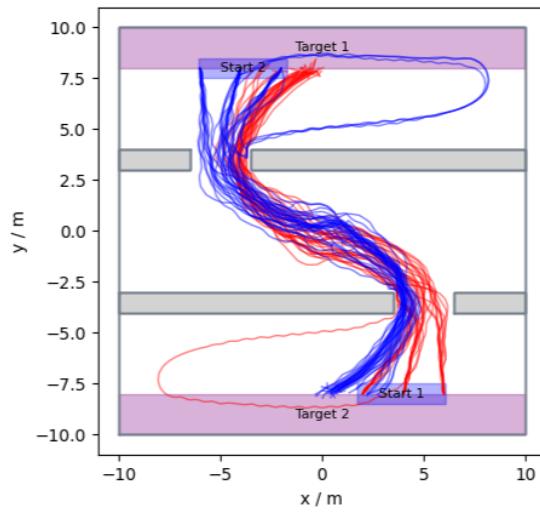
Ancora

Nell’ambiente “Ancora”, le traiettorie degli agenti mostrano un elevato livello di coordinazione. I due gruppi di agenti navigano attraverso i corridoi e si incontrano all’angolo cieco senza causare congestioni.



Doppia Porta Opposta

Nell'ambiente “Doppia Porta Opposta”, la maggior parte dei pedoni riesce a mantenere una buona traiettoria durante l'esecuzione, mentre una piccola parte, a causa di un'elevata densità ai passaggi, crea una diramazione e si allontana, mostrando un comportamento non ottimale. Inoltre, al centro dell'incrocio, non vi è un buon coordinamento tra agenti, in quanto i due flussi non sono ben delineati.



(e) Traiettoria: Doppia Porta Opposta

Capitolo 3

Descrizione del progetto software

In questo capitolo verranno analizzati la struttura e il funzionamento del progetto, al fine di fornire una comprensione più dettagliata della sua organizzazione e operatività. Per una comprensione completa, è possibile consultare il repository del progetto open source realizzato e rilasciato sotto licenza MIT¹. Il codice sorgente, la documentazione dettagliata e gli esempi possono essere visualizzati e scaricati dal repository, consentendo così una valutazione approfondita e la possibilità di contribuire allo sviluppo continuo del modello.

3.1 Componenti Godot

3.1.1 Training Scene

La **Training Scene** è la scena fondamentale che gestisce l’addestramento del modello. Il suo compito principale è eseguire il cambio di scena tra un ambiente di training e il successivo, mentre le condizioni per la transizione sono gestite direttamente lato **Python**. Al suo interno è memorizzato un array contenente gli ambienti di training, specificati nell’ordine in cui devono essere eseguiti. Per eseguire la fase di retraining, la **Retraining Scene** deve essere inserita come ultimo elemento in questo array.

3.1.2 Level Batch

Il **Level Batch** è la scena che istanzia in parallelo i livelli su cui addestrare il modello. Contiene il livello da istanziare e il numero di istanze da eseguire in parallelo. Al termine dell’esecuzione del livello corrente, il **Level Batch** notifica la **Training Scene** per effettuare il cambio di scenario.

¹<https://github.com/Ruben-2828/RL-Godot-Pedestrian-Simulation.git>

3.1.3 Sync

Il nodo **Sync**, aggiunto dalla libreria `godot-rl-agents`, gestisce la comunicazione con **Python**. Le sue funzioni principali sono:

1. Sincronizzarsi con il lato **Python**.
2. Inviare le informazioni relative all’ambiente corrente.
3. Ricevere le azioni scelte dal modello e impostarle ai pedoni corrispondenti.
4. Inviare i reward ottenuti in seguito all’esecuzione delle azioni ricevute.
5. Segnalare la fine del livello corrente quando riceve il messaggio di terminazione da **Python**.

La comunicazione tra **Python** e **Godot** avviene tramite lo scambio di messaggi in formato **JSON** su uno stream **TCP**.

3.1.4 Level Manager

Il **Level Manager** è il nodo che gestisce i singoli livelli. Contiene il livello di cui si occupa e le sue funzioni principali sono:

1. Inizializzare il livello e tutti i suoi elementi (pedoni, target, randomizzazione, ecc.).
2. Notificare la fine di un episodio di training.

3.1.5 Pedestrian

Il **Pedestrian** è l’elemento centrale di questa architettura, incaricato di imparare a orientarsi e spostarsi nei vari ambienti. Si occupa di eseguire le azioni ricevute da **Python** e di calcolare i reward relativi a quelle azioni. I suoi componenti principali sono:

1. Il pedone stesso, che esegue le azioni ricevute (muoversi in avanti e ruotare).
2. L’**AIController**, che riceve le azioni e trasmette i reward e le osservazioni.
3. I sensori, che rilevano ostacoli, target e altri pedoni, computando i reward e le osservazioni.

AIController

L’**AIController** è il nodo che colleziona le osservazioni del pedone e imposta le azioni che deve eseguire. Durante la fase di collezione delle osservazioni, l’**AIController** contatta i sensori del pedone per ottenerle e le invia al nodo **Sync**. Quando deve impostare le azioni del pedone, l’**AIController** riceve le azioni dal nodo **Sync** e le trasmette al pedone.

3.2 Componenti Python

3.2.1 Runner

Il **Runner** è il componente principale per l’addestramento del modello. Gestisce tutte le operazioni necessarie per completare il training. Le sue funzioni principali sono:

1. Caricare i livelli di training tramite il **ConfigParser**.
2. Eseguire ogni livello specificato nelle configurazioni.
3. Verificare se i criteri di terminazione di un livello sono stati raggiunti e notificare **Godot** in caso affermativo.
4. Mantenere un log delle transizioni degli ambienti.
5. Esportare il modello **ONNX** al termine dell’addestramento.

3.2.2 Config

Un elemento fondamentale del progetto è la configurazione lato **Python**. Questa si divide in tre file principali:

1. **Config File**: necessario per specificare gli iperparametri del modello.
2. **Curriculum File**: necessario per specificare gli ambienti del curriculum e i relativi valori associati.
3. **Config Parser**: necessario per validare i dati inseriti.

Curriculum File

Il **Curriculum File** è un file **YAML** che contiene tutte le configurazioni degli ambienti necessarie per l’esecuzione del training. Ogni livello del curriculum è definito con 3 parametri specifici:

1. **mean_reward**: soglia da raggiungere per considerare un livello superato
2. **episode_mean**: numero di episodi sulla quale eseguire la media
3. **cycles**: numero di cicli per la quale è possibile continuare l'esecuzione

Di seguito è riportato il file del curriculum utilizzato durante il modello base del progetto.

Codice 3.1: Curriculum File

```

1 Curriculum:
2   StartEZ:
3     mean_reward: 5.7
4     episode_mean: 36
5     cycles: 10
6   Start:
7     mean_reward: 3.0
8     episode_mean: 36
9     cycles: 5
10  Observe:
11    mean_reward: -24.0
12    episode_mean: 36
13    cycles: 5
14  StartCorridor:
15    mean_reward: 4.5
16    episode_mean: 36
17    cycles: 15
18  Turns:
19    mean_reward: 10.0
20    episode_mean: 36
21    cycles: 10
22  TurnObstacles:
23    mean_reward: 10.0
24    episode_mean: 36
25    cycles: 10
26  NarrowDoor:
27    mean_reward: 5.0
28    episode_mean: 36
29    cycles: 15
30  Corridor3vs3:
31    mean_reward: 4.0
32    episode_mean: 36
33    cycles: 15
34  T-Junction:
35    mean_reward: 4.0
36    episode_mean: 36
37    cycles: 5
38  Intersection4:
39    mean_reward: 5.0
40    episode_mean: 36
41    cycles: 15

```

```
42  NarrowDoorDouble:  
43      mean_reward: 4.0  
44      episode_mean: 36  
45      cycles: 15
```

Config File

Il **Config File** è un file YAML che contiene tutte le configurazioni del modello necessarie per l'esecuzione del training. Di seguito è riportato il file della configurazione del modello base del progetto.

Codice 3.2: Config File

```
1 hyperparameters:  
2     learning_rate: 0.0003  
3     n_steps: 32  
4     batch_size: 320  
5     n_epochs: 3  
6     gamma: 0.99  
7     gae_lambda: 0.95  
8     clip_range: 0.2  
9     clip_range_vf: null  
10    normalize_advantage: True  
11    ent_coef: 0.0001  
12    vf_coef: 0.5  
13    max_grad_norm: 0.5  
14    use_sde: False  
15    sde_sample_freq: -1  
16    target_kl: null  
17    stats_window_size: 1  
18    device: "auto"  
19    verbose: 0  
20    policy_kwargs:  
21        net_arch:  
22            - 256  
23            - 256  
24 max_steps: 1000000000000000000  
25 retraining_steps: 200_000
```

Config Parser

Il **Config Parser** è una classe Python utilizzata per caricare e validare i dati inseriti all'interno del **Config File** e del **Curriculum File**.

3.2.3 Callbacks

Le callbacks sono il principale metodo utilizzato in questa architettura per interrompere l'addestramento del modello. Ci sono due tipi di callback:

1. `EndTrainingOnMeanRewardReachedCallback` per fermare l'addestramento nel momento in cui le condizioni di terminazione sono state raggiunte
2. `EndTrainingOnEarlyFailCallback` per fermare interamente la fase di training quando le prestazioni del modello sono troppo scarse.

Per effettuare una valutazione della prima callback viene eseguita una media sfondata dei reward ogni 36 episodi. Se questa soddisfa i requisiti specificati nel file delle configurazioni del curriculum, il `Runner` interrompe l'ambiente corrente e istanzia il livello successivo sincronizzandosi con il nuovo ambiente lato `Godot`. Se dopo un certo numero di cicli prestabiliti la media non è ancora stata raggiunta, subentra la seconda callback che ferma interamente il training terminando l'intera esecuzione dello script `Python`.

3.3 Flusso di esecuzione

In questa sezione viene spiegato a grandi linee il normale flusso di esecuzione del programma. Per facilitarne la comprensione, il flusso è presentato in diverse sottosezioni, ciascuna delle quali descrive una parte specifica del processo.

3.3.1 Training Scene

La prima sottosezione riguarda la `Training Scene` con il `Level Batch`. All'avvio dell'esecuzione del progetto su `Godot`, lo script della `Training Scene` entra in un loop in cui istanzia il `Level Batch` del primo livello presente nell'array dei livelli.

Una volta che il primo livello terminerà, verrà istanziato il successivo e così via, fino all'ultimo livello: la scena di `Retraining`. Quando anche questa sarà terminata, verrà creato, lato `Python`, un modello in formato `ONNX` assegnabile al nodo `Sync` della `Testing Scene`.

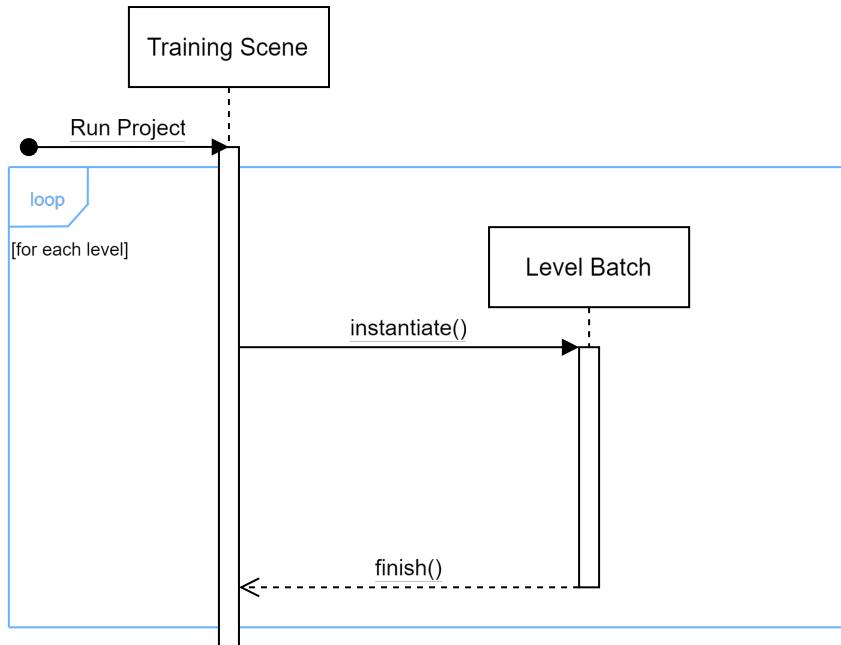


Figura 3.1: Diagramma di Sequenza: Training Scene

3.3.2 Level Batch

In questa fase, il **Level Batch** entra a sua volta in un loop in cui istanzia tanti **Level Manager** quanti specificati nella costante `batch_size`. Questo approccio consente l’addestramento parallelo, accelerando il processo di apprendimento.

Dopo aver istanziato i **Level Manager**, il **Level Batch** svolge un altro compito fondamentale: l’abilitazione del Reinforcement Learning. A tale scopo, il **Level Batch** imposta il nodo `Sync` a `ready`. Il nodo `Sync` si mette in ascolto con `Python` e, una volta stabilita la comunicazione, avviene lo scambio di messaggi. Fino a quando `Python` non invia un messaggio di tipo `close`, il nodo `Sync` continuerà a ricevere l’azione da intraprendere e a restituire il risultato conseguente.

Quando il messaggio di chiusura arriva al nodo `Sync`, questo termina la comunicazione con il **Level Batch**, che a sua volta interrompe la propria esecuzione, permettendo alla **Training Scene** di istanziare il livello successivo.

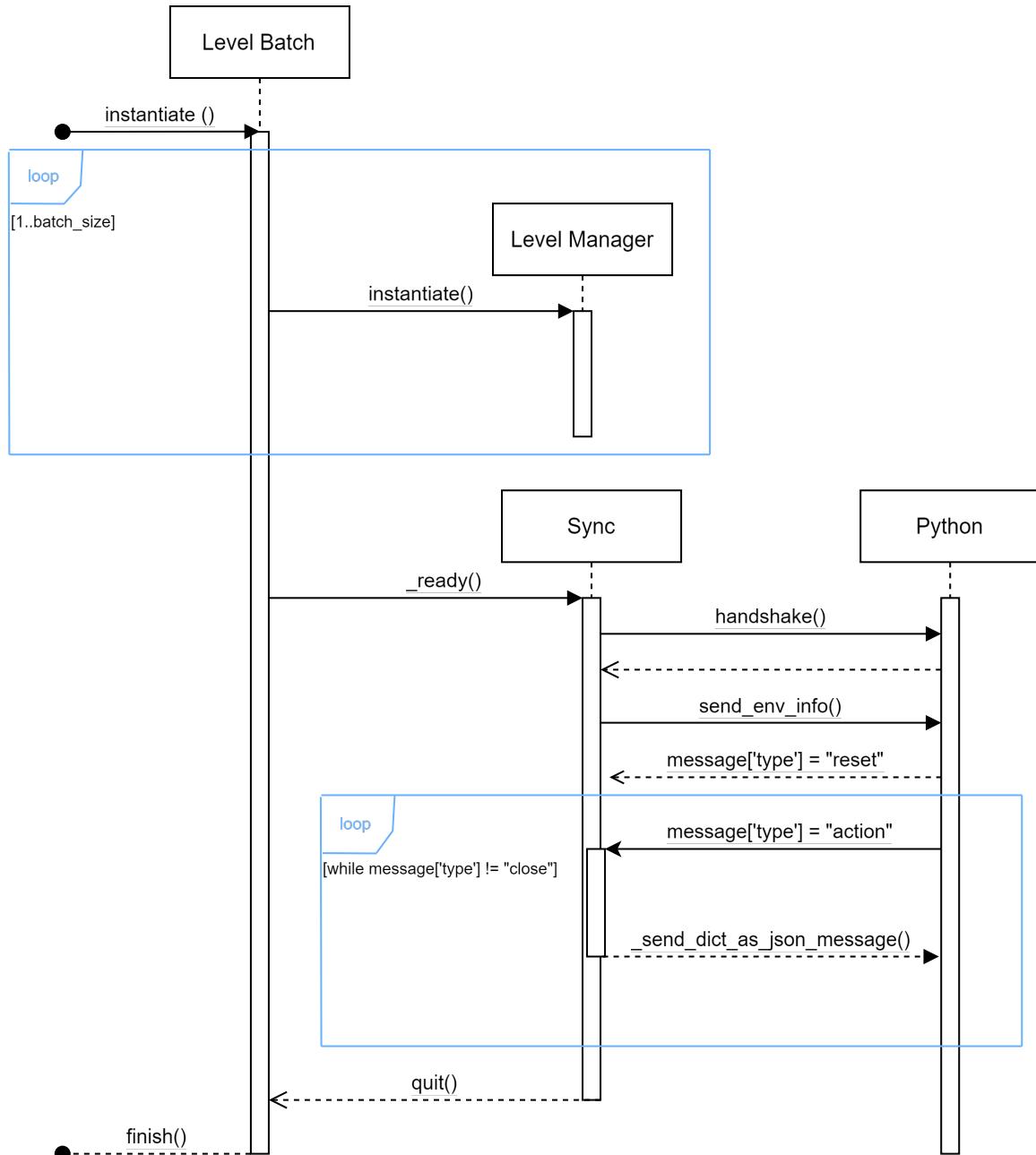


Figura 3.2: Diagramma di Sequenza: Level Batch

3.3.3 Level Manager

Il **Level Manager** è il nodo responsabile della gestione di tutti gli elementi all'interno di un singolo livello del curriculum. La prima operazione che esegue è l'istanziazione dell'effettivo livello, creando l'**environment** su cui eseguire il training. Una volta istanziato il livello, il **Level Manager** imposta a **ready** il nodo **PedestrianController**, che controlla tutti i pedoni presenti nella scena. I pedoni vengono istanziati e iniziano ad eseguire le proprie azioni.

Quando i pedoni terminano la loro esecuzione, ovvero quando tutti raggiungono il target finale o scade il numero massimo di timesteps predisposti per l'**environment**, vengono resettati dal **Pedestrian Controller**. Quest'ultimo notifica inoltre se un episodio è terminato. La terminazione dell'intero livello è gestita da **Python**, che verifica se il **Mean Reward**, definito nel file di configurazione del curriculum, è stato raggiunto, oppure se il training non è stato sufficientemente efficace, attivando la **Callback Early Fail**.

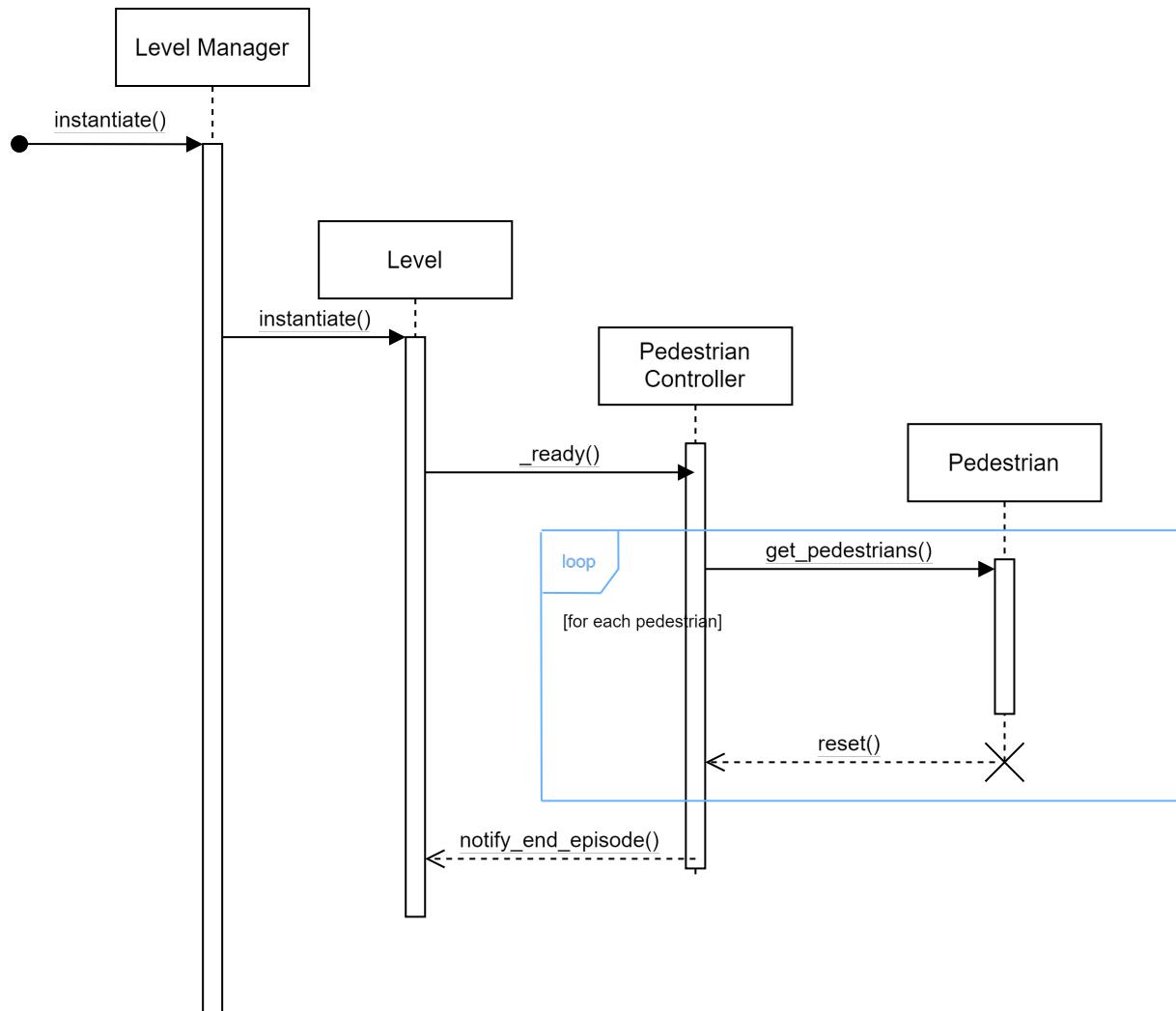


Figura 3.3: Diagramma di Sequenza: Level Manager

Capitolo 4

Analisi ad Alta Densità

4.1 Introduzione

In questo capitolo, presentiamo un modello progettato specificamente per esaminare il comportamento degli agenti in ambienti caratterizzati da alta densità. A differenza del modello precedente, che si limitava a valutare l'efficacia con cui gli agenti raggiungevano le uscite degli ambienti, questo nuovo approccio mira a esplorare in modo più approfondito il comportamento e il coordinamento degli agenti di fronte a situazioni di traffico intenso.

4.2 Glossario

In questa sezione, si presenta un'estensione al glossario precedentemente fornito nel Capitolo 2, aggiungendo definizioni utili per una comprensione approfondita dei concetti trattati in tale capitolo.

I colli di bottiglia (“bottleneck”, in inglese) rappresentano situazioni in cui il flusso di movimento è limitato, causando un accumulo di persone o oggetti in spazi ristretti. Questo fenomeno può manifestarsi in diversi contesti, come ingressi stretti, corridoi angusti o punti di accesso limitati. L’analisi dei colli di bottiglia è essenziale per comprendere il comportamento del flusso in condizioni di alta densità e per migliorare la progettazione di spazi pubblici e infrastrutture, garantendo sicurezza ed efficienza nei movimenti.

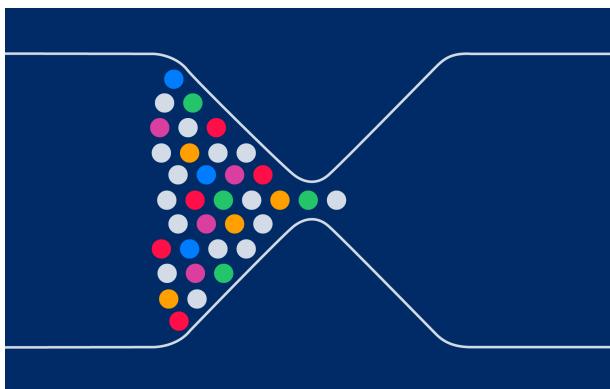


Figura 4.1: Rappresentazione di un bottleneck

I **poligoni di Voronoi** costituiscono una partizione del piano in regioni dove ogni regione è associata a uno specifico punto, noto come sito o generatore. Ogni punto nel piano appartiene alla regione di Voronoi del sito più vicino, il che significa che la regione è costituita da tutti i punti del piano che sono più vicini a quel sito rispetto a qualsiasi altro generatore nella partizione. La Figura 4.2 mostra una utilizzo in contesti reali dei poligoni di Voronoi¹.

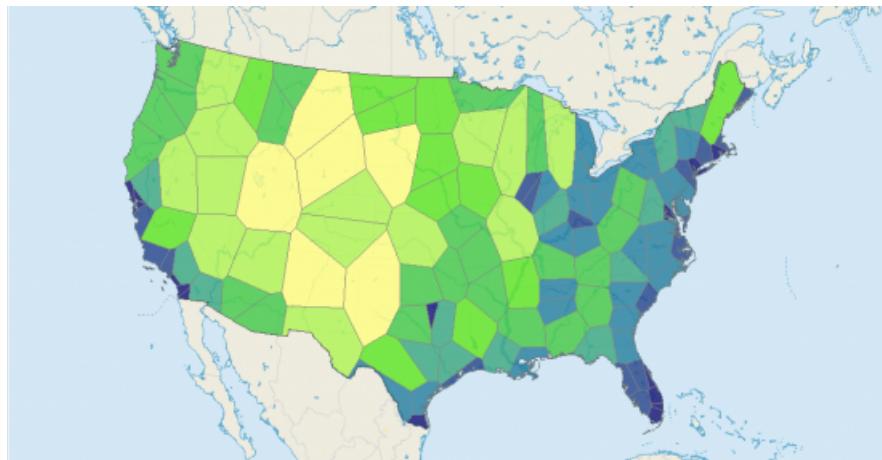


Figura 4.2: Visualizzazione degli aeroporti più remoti degli USA tramite poligoni di Voronoi

La **densità di Voronoi** è una misura della densità dei punti di controllo all'interno dei poligoni di Voronoi. Essenzialmente, è una misura della quantità di punti di controllo presenti in una determinata area o regione dello spazio definita dai poligoni di Voronoi. La densità di Voronoi con **cutoff** è una variante della densità di Voronoi che tiene conto solo dei punti di controllo situati entro una certa distanza, definita cutoff, da un punto di interesse. Il cutoff è essenzialmente un cerchio approssimato che dà la massima estensione di un singolo poligono di Voronoi. In Figura 4.3 una rappresentazione della densità di Voronoi con cutoff².

¹<https://gisgeography.com/voronoi-diagram-thiessen-polygons/>

²https://pedpy.readthedocs.io/en/stable/user_guide.html

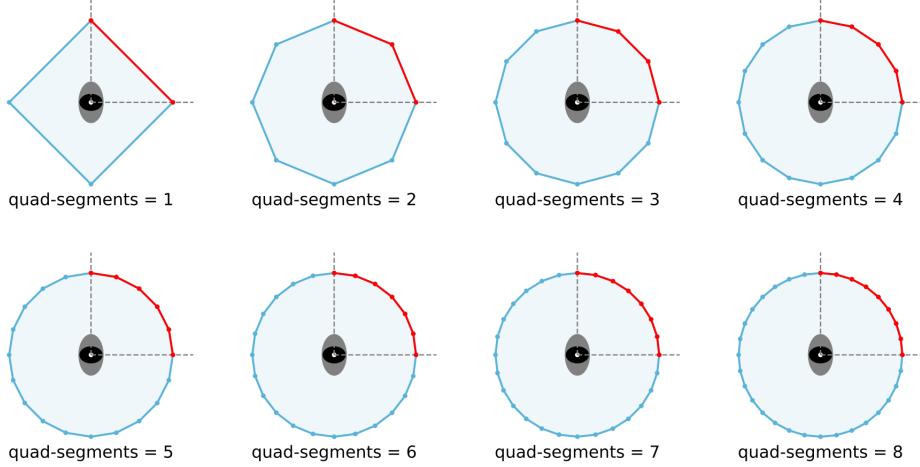


Figura 4.3: Rappresentazione dei cutoff al variare del numero di segmenti

I **profili di densità** sono strumenti utilizzati in varie discipline, come la fisica, l'ingegneria del traffico e la dinamica delle folle, per descrivere e analizzare la distribuzione spaziale di un gruppo di entità (ad esempio, particelle, veicoli o pedoni) in una determinata area. Sono stati utilizzati 3 tipi di profili di densità:

1. **Profilo di densità di Voronoi:** in ogni cella, la velocità di Voronoi v_{voronoi} è definita come

$$v_{\text{voronoi}}(t) = \frac{\int(\int v_{xy} dx) dy}{A(M)}$$

dove $v_{xy} = v_i$ è la velocità individuale di ciascun pedone, con $V_i \cap M$ e $A(M)$ l'area della cella della griglia.

2. **Profilo di densità classico:** in ogni cella, la densità ρ_{classic} è definita da

$$\rho_{\text{classic}} = \frac{N}{A(M)},$$

dove N è il numero di pedoni all'interno della cella della griglia M e $A(M)$ è l'area di quella cella della griglia.

3. **Profilo di densità gaussiana:** in ogni cella, la densità ρ_{gaussian} è definita da

$$\rho_{\text{gaussian}} = \sum_{i=1}^N \delta(r_i - c),$$

dove r_i è la posizione di un pedone e c è il centro della cella della griglia. Infine, $\delta(x)$ è approssimata da una Gaussiana

$$\delta(x) = \frac{1}{\sqrt{\pi}a} \exp \left[-\frac{x^2}{a^2} \right].$$

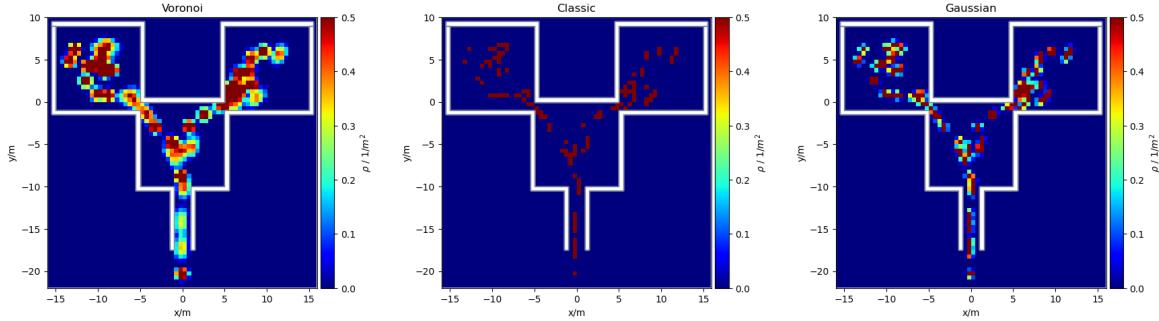


Figura 4.4: Esempio dei 3 tipi di densità utilizzati all'interno dello studio

4.3 Casi di studio

Per sviluppare l'analisi in questione sono stati presi in considerazione due studi.

4.3.1 Analisi della larghezza del corridoio prima di un bottleneck

Lo studio [1] **Crowds in front of bottlenecks at entrances from the perspective of physics and social psychology** esplora l'effetto di diversi fattori sul comportamento delle folle nei bottleneck. Nella nostra analisi, ci siamo concentrati principalmente sull'impatto della larghezza del corridoio d'ingresso sul flusso e sulla densità dei pedoni. I risultati evidenziano che, all'aumentare della larghezza del corridoio, vi è un aumento della densità di persone per metro quadrato.

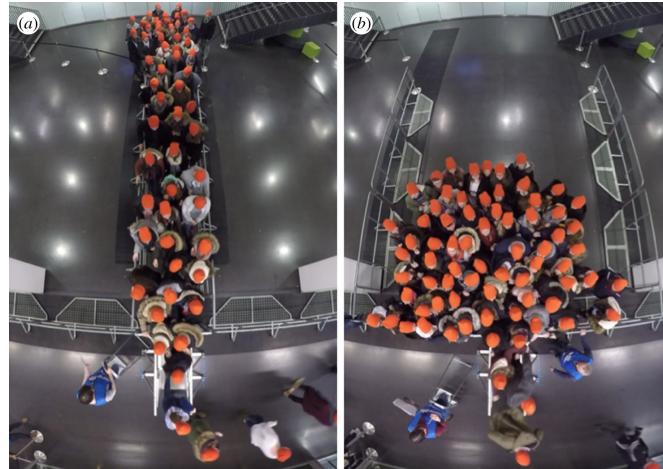


Figura 4.5: Rappresentazione di due diverse ampiezze del corridoio prima di un bottleneck da [1]

4.3.2 Confronto dei tempi di evacuazione di collegamenti centrali e ad angolo

Concentrandoci sulla sezione “Merging” dello studio [8] **Pedestrian flow through multiple bottlenecks**, esploriamo gli effetti paradossali dei progetti architettonici sul flusso pedonale. Lo studio mette a confronto due tipi di collegamenti, centrale e ad angolo, e rivela che nonostante i collegamenti ad angolo migliorano i tempi di evacuazione locale, aumentano paradossalmente il tempo di evacuazione totale a causa dei tassi di afflusso più elevati che portano ad una maggiore densità vicino alle uscite.

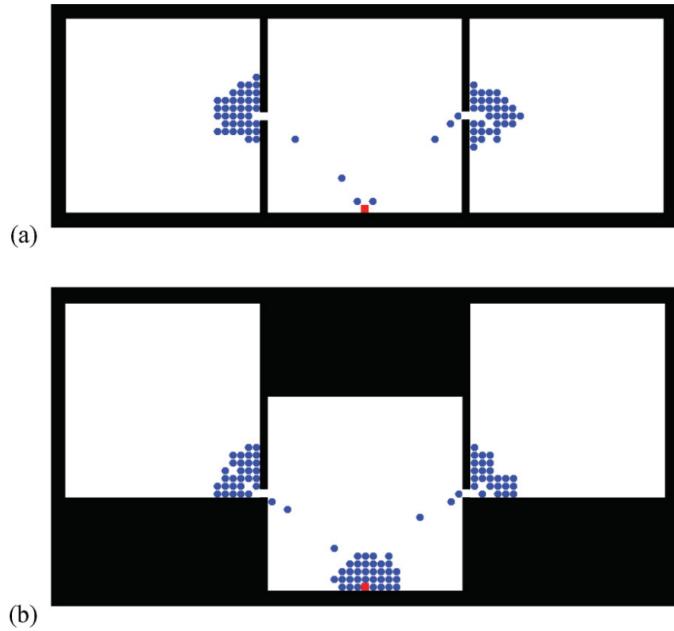


Figura 4.6: Rappresentazione del flusso con collegamento al centro e ad angolo da [8]

4.4 Modello ad alta densità

4.4.1 Test su modello base

Una volta terminata la realizzazione del modello base, abbiamo ritenuto necessario valutare se esso fosse sufficientemente addestrato per garantire prestazioni ottimali in situazioni caratterizzate da un elevato traffico pedonale. A tal fine, abbiamo incrementato il numero di pedoni presenti in alcuni ambienti specifici, come "Doppia Porta Opposta" e "Ancora", e successivamente abbiamo eseguito una fase di test in tali contesti.

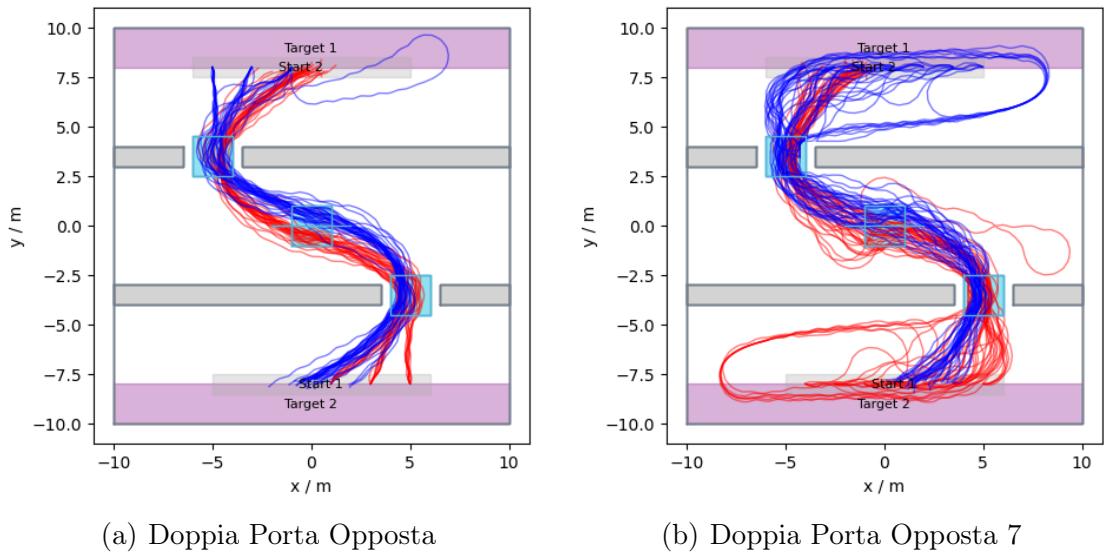


Figura 4.7: Confronto tra le traiettorie in “Doppia Porta Opposta” e “Doppia Porta Opposta 7”

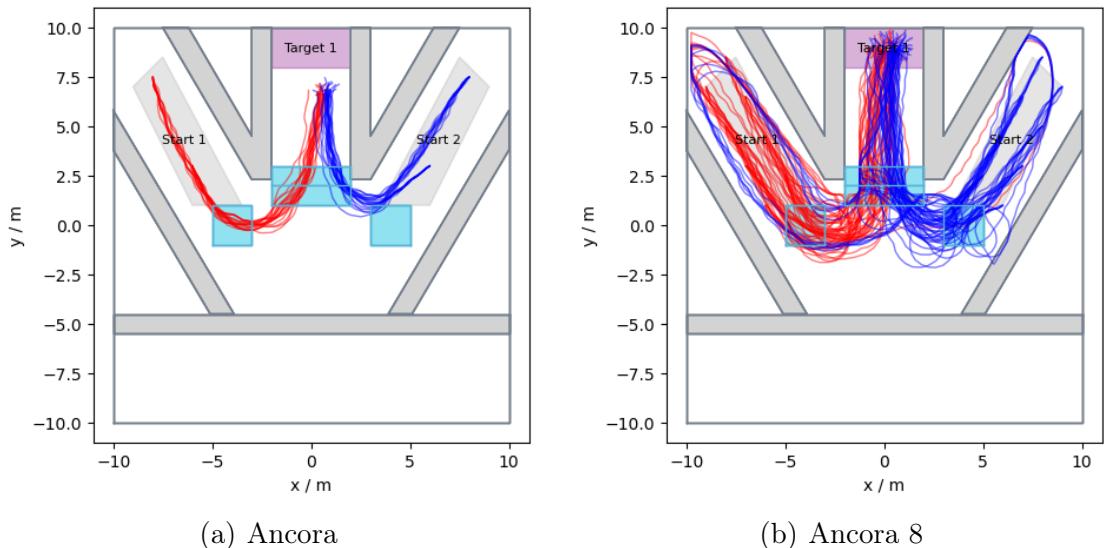


Figura 4.8: Confronto tra le traiettorie in “Ancora” e “Ancora 8”

I risultati ottenuti, sia dall’analisi delle esecuzioni sia dalla visualizzazione dei dati tramite grafici, hanno chiaramente indicato che il modello base non era adeguatamente addestrato per gestire situazioni ad alto traffico. In effetti, esso non è riuscito a garantire una gestione efficiente dei nuovi ambienti di test. Questi risultati suggeriscono la necessità di un miglioramento del modello per affrontare con successo scenari caratterizzati da un elevato numero di pedoni.

4.4.2 Ambienti

Per migliorare la familiarità del modello con situazioni di traffico più intenso rispetto al modello base, sono state apportate diverse modifiche ad ambienti precedentemente realizzati e sono stati introdotti due nuovi ambienti. Di seguito una tabella riassuntiva dei nuovi ambienti di addestramento. Successivamente, verranno mostrati nello specifico gli ambienti introdotti e quelli modificati.

Comportamento	Ambiente	Retraining
Camminare verso il target	StartEz	No
	Start	Sì
Guardare il target	Osserva	Sì
Navigare in corridoi stretti	StartCorridoio	No
	Corridoio3	Sì
Curvare e evitare prossemica muri	Curve	No
Curvare e evitare prossemica muri con altri agenti	CurveOstacoli3	Sì
Uscire da una stanza	UscitaStanza	Sì
Evitare agenti nella stessa direzione	PortaStretta12	Sì
Evitare agenti in direzioni diverse	Corridoio3vs3	Sì
	Intersezione4	Sì
	T3	Sì
Combinazione di tutti i comportamenti	DoppioPortaStretta12	Sì

Tabella 4.1: Ambienti d’addestramento

Porta Stretta 12

La larghezza del passaggio è stata ridotta a 1.5 metri e le dimensioni della stanza sono state ridotte a 12x12 metri. Queste modifiche aumentano la densità di pedoni nell’ambiente e rendono più impegnativo il superamento della porta centrale.

Doppio Porta Stretta 12

Eseguita la stessa modifica dell’ambiente Porta Stretta 12.



(a) Porta Stretta 12

(b) Doppio Porta Stretta 12

Figura 4.9: Ambienti “Porta Stretta 12” e “Doppio Porta Stretta 12” del curriculum ad alta densità.

Curve Ostacoli 3

Il passaggio da un singolo agente è stato trasformato in un gruppo di 3 pedoni. Questo permette al modello di affrontare situazioni più complesse in cui è necessario coordinare il movimento di più agenti contemporaneamente.

Giunziona a T 3

I passaggi sono stati ridotti da 3 metri a 2 metri e è stato aggiunto un pedone per gruppo. Questa modifica rende l’attraversamento dell’incrocio più complesso e richiede una migliore coordinazione tra gli agenti.

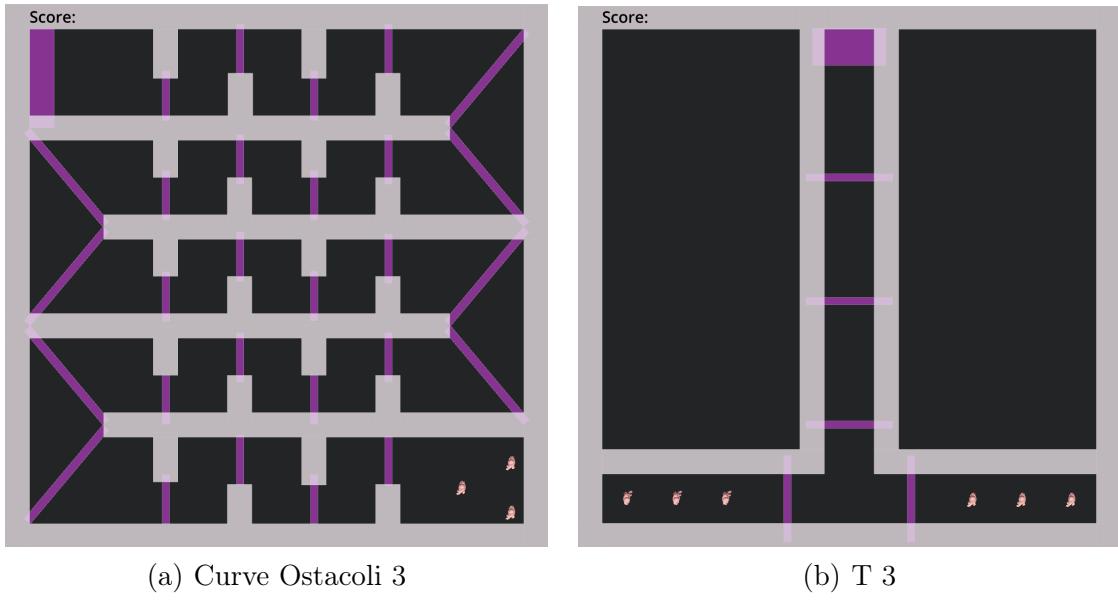


Figura 4.10: Ambienti “Curve Ostacoli 3” e “T3” del curriculum ad alta densità.

Corridoio 3

Questo nuovo ambiente è una semplificazione di Corridoio 3v3, ma presenta un unico gruppo di pedoni. Ciò aiuta il modello a imparare a gestire la vicinanza tra i pedoni ed evitare i muri senza interferenze da altri flussi di traffico.

Uscita Stanza

Questo nuovo ambiente è stato creato per far apprendere all’agente a raggiungere un’uscita posizionata all’angolo della stanza. Questo scenario introduce una nuova sfida nell’ottimizzazione del percorso verso l’uscita.

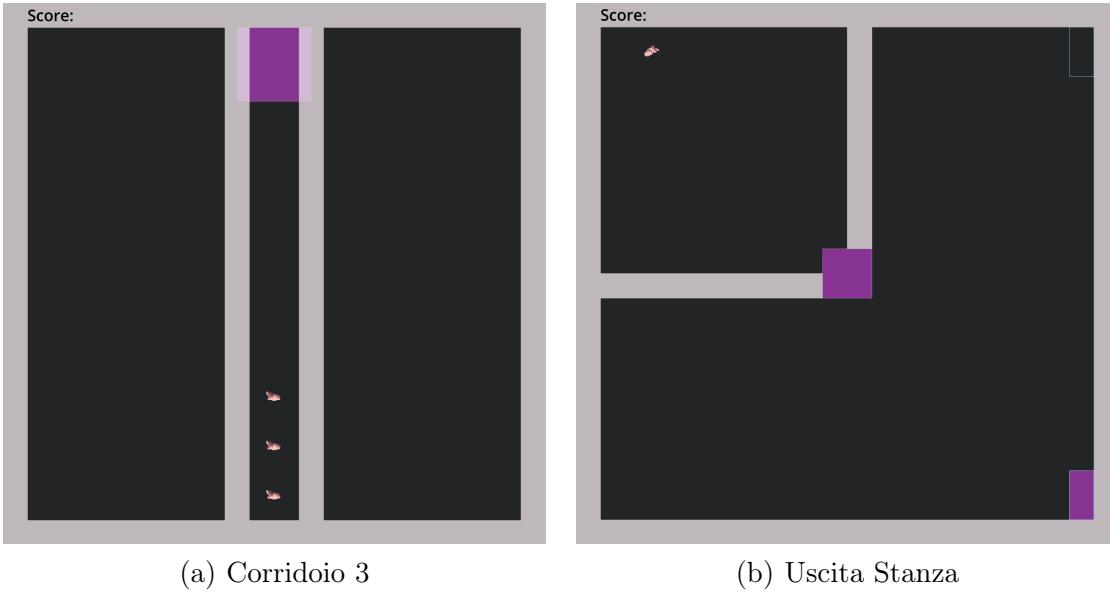


Figura 4.11: Ambienti “Corridoio 3” e “Uscita Stanza” del curriculum ad alta densità.

4.4.3 Rewards

Oltre alle modifiche agli ambienti, è stato necessario apportare leggere modifiche al modello di reinforcement learning utilizzato. In particolare, sono stati rimossi i reward negativi relativi alla prossemica a lunga e media gittata. Questa decisione è stata presa poiché tali valori si sono dimostrati troppo restrittivi in situazioni ad alta densità. I precedenti reward spingevano i pedoni a evitare gli altri invece di formare una fila per raggiungere un’uscita, il che non rifletteva adeguatamente le dinamiche reali in situazioni ad alto traffico.

L’eliminazione di questi reward negativi ha permesso di simulare meglio il comportamento dei pedoni in scenari affollati, dove la formazione di file ordinate è un aspetto cruciale per una gestione efficiente del traffico pedonale.

4.5 Andamento dell’addestramento

L’andamento dell’addestramento è visualizzabile attraverso la media del reward ottenuto durante il processo, come avviene per il modello base dell’esperimento.

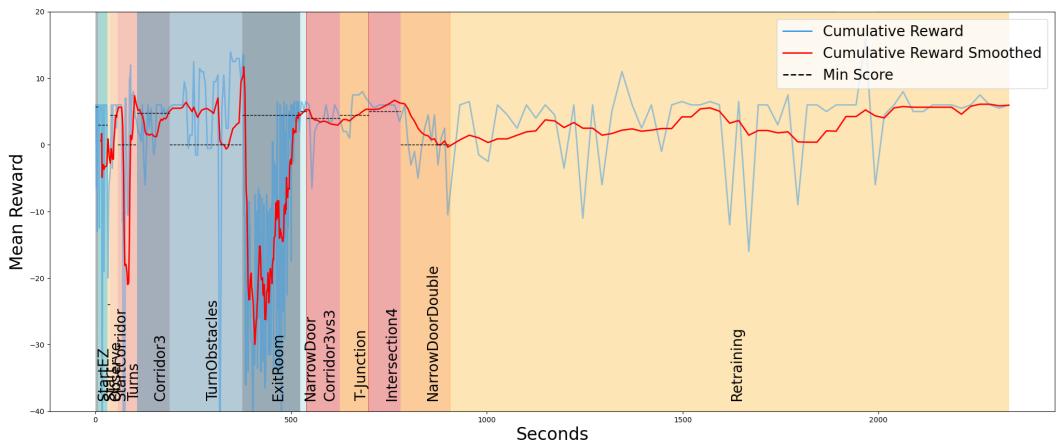


Figura 4.12: Andamento del reward all'interno degli ambienti di training

Un'altra visualizzazione utile è rappresentata dalla lunghezza degli episodi negli ambienti. I nuovi livelli introdotti, “Uscita Stanza” e “Corridor 3”, richiedono tempi nella media per essere completati, nonostante la grossa perdita di reward alle prime ripetizioni dell’ambiente “Uscita Stanza”. La riduzione della dimensione delle stanze ha anche permesso ad ambienti che solitamente richiedevano più tempo di terminare più rapidamente. Queste due modifiche hanno consentito di mantenere la durata totale dell’addestramento vicina a quella del modello base, come possiamo osservare in Figura 4.13.

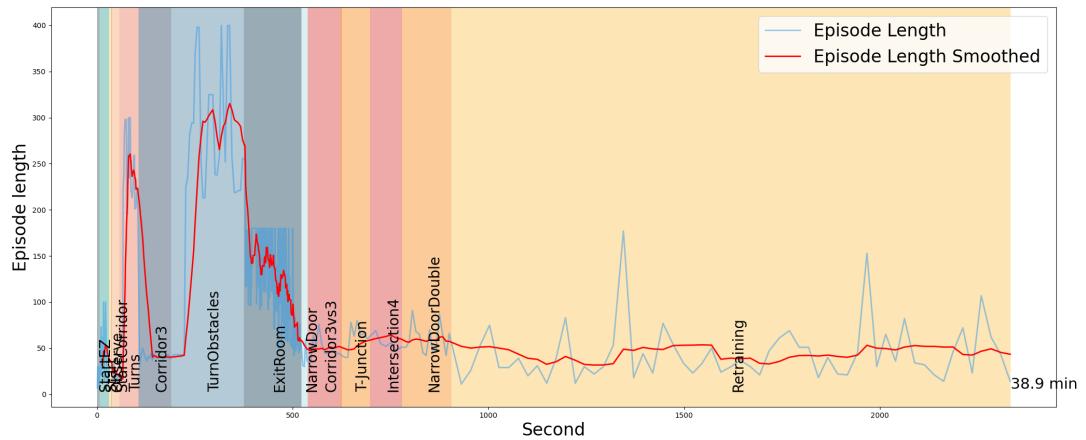


Figura 4.13: Durata degli ambienti di training

4.6 Test

La fase di test del modello esteso ha lo scopo di valutare quanto l’agente abbia appreso i nuovi comportamenti necessari per operare in scenari ad

alta densità, mantenendo al contempo i comportamenti acquisiti durante l’addestramento con il modello base.

A tal fine, la fase di test è stata suddivisa in tre parti:

1. **Modello base:** Si è verificato che vengano mantenuti i comportamenti del modello base.
2. **Modello aumentato:** È stato valutato se i pedoni hanno imparato a gestire adeguatamente scenari ad alto traffico e con interazioni sociali complesse, utilizzando i precedenti ambienti ma aumentati nel numero di pedoni.
3. **Modello esteso:** È stato valutato se i pedoni allenati tramite reinforcement learning mantengono comportamenti simili a quelli umani in casi di test specifici.

Ambienti di Test

Per testare i comportamenti sul modello base, sono stati utilizzati gli ambienti “Ancora” e “Doppia Porta Opposta”, riproposti in Figura 4.14.

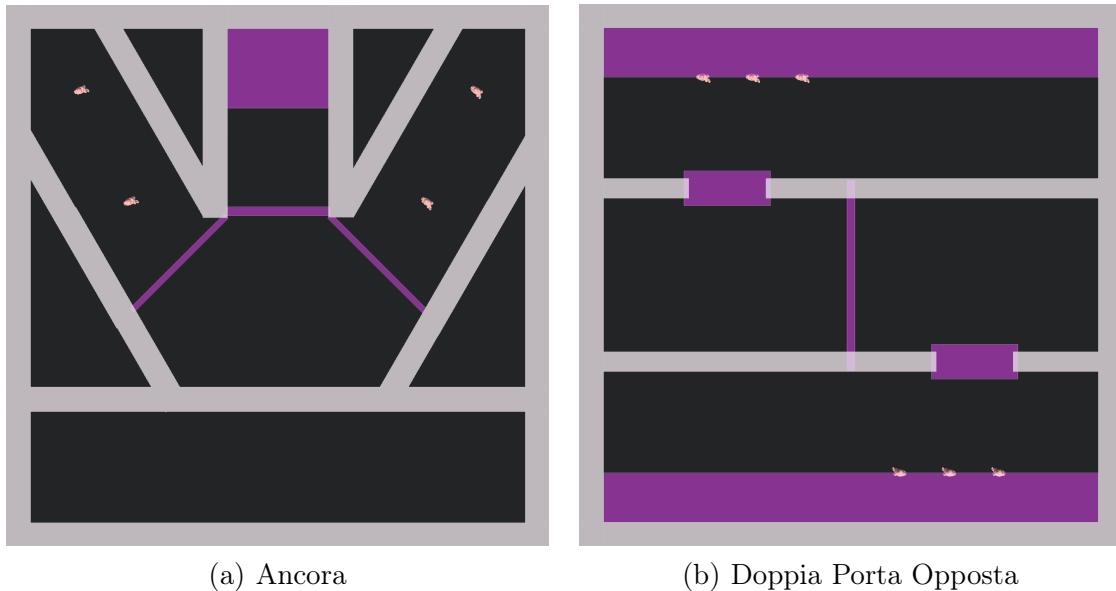


Figura 4.14: Ambienti “Ancora” e “Doppia Porta Opposta” della fase di test del “modello base”.

Per testare i comportamenti sul modello aumentato, sono stati utilizzati le stesse versioni degli ambienti ma aumentati nel numero di pedoni, presentando i nuovi ambienti “Ancora 8”, con due gruppi da otto pedoni, e “Doppia Porta Opposta 7” con due gruppi da sette pedoni, osservabili in Figura 4.15.

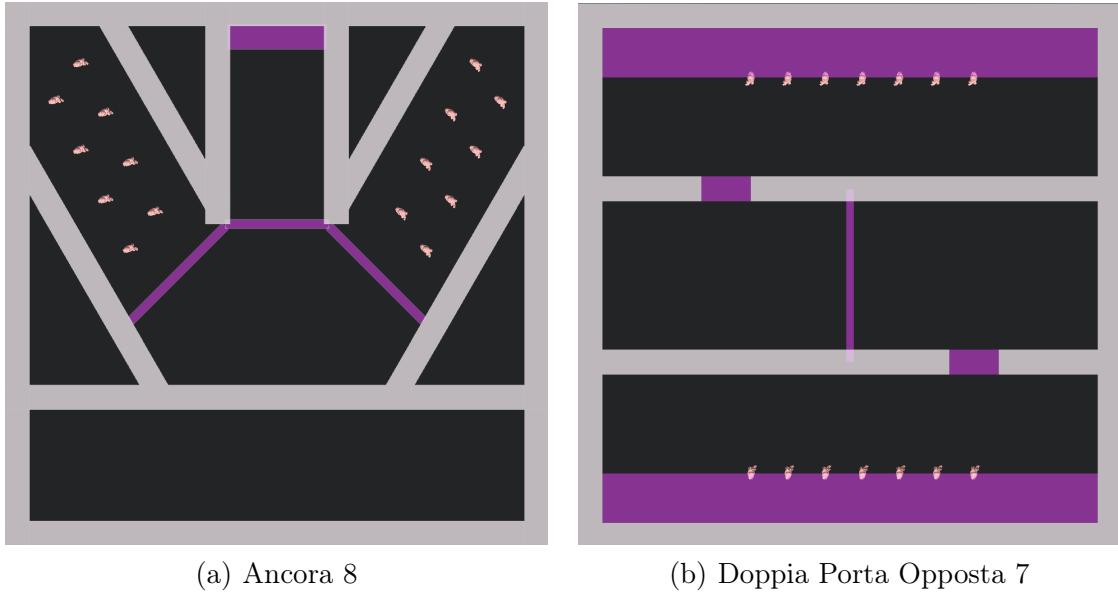


Figura 4.15: Ambienti “Ancora 8” e “Doppia Porta Opposta 7” della fase di test del “modello aumentato”.

Per testare i comportamenti sul modello esteso, sono stati replicati gli ambienti utilizzati negli studi citati. Per l’analisi della folla di fronte a un collo di bottiglia, sono state create tre versioni dello stesso ambiente, ognuna con 60 agenti posizionati davanti a un corridoio di larghezza diversa: il più ampio ha una larghezza di 5.6 metri, la versione media 3.5 metri e il più stretto 2 metri. Al termine del collo di bottiglia, un corridoio conduce il pedone all’uscita dell’ambiente.

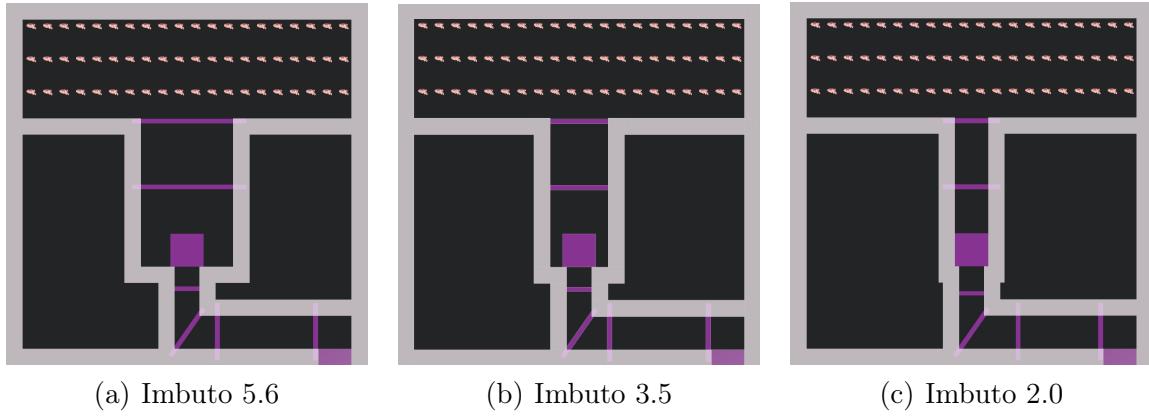


Figura 4.16: Ambienti “Imbuto 5.6”, “Imbuto 3.5” e “Imbuto 2.0” della fase di test del “modello esteso”.

Per l’analisi dei tempi di evacuazione di strutture con porte posizionate agli angoli o al centro delle stanze, sono stati creati due ambienti. Questi ambienti sono sostanzialmente simili: entrambi includono due gruppi di 20 pedoni posizionati in due stanze separate, per un totale di 40 pedoni per

ambiente. Una stanza centrale collega le due stanze con gli agenti, e da questa stanza si estende un corridoio che i pedoni devono percorrere per completare il livello con successo. L'unica differenza tra i due ambienti è la posizione delle porte: in Figura 4.17(b), le porte sono situate al centro delle pareti e quindi “parallele” alla stanza che le congiunge, mentre in Figura 4.17(a), le porte sono posizionate agli angoli delle stanze. Di seguito sono illustrati i due ambienti.

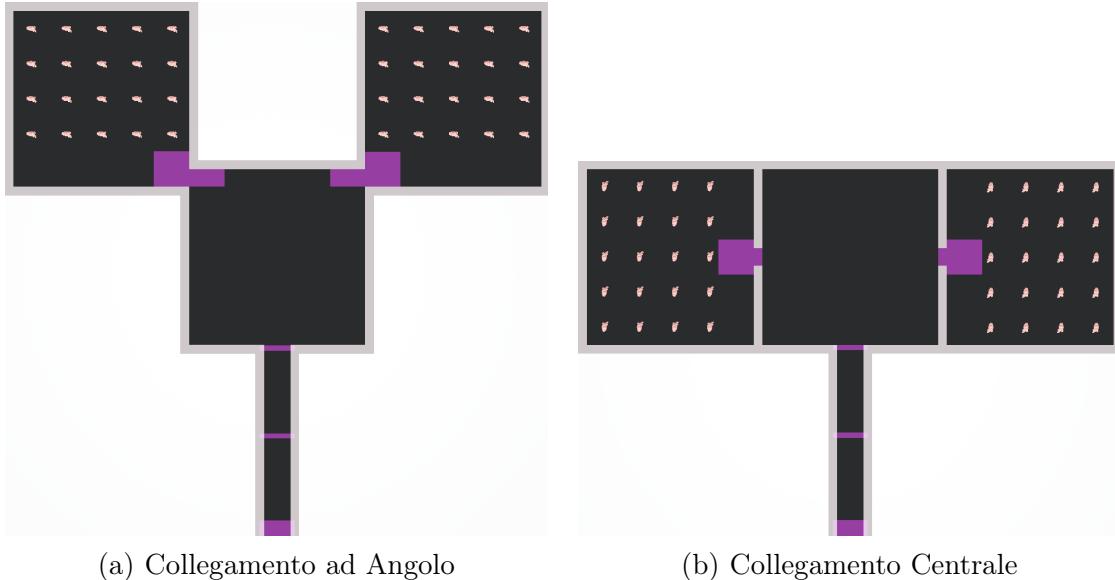


Figura 4.17: Ambienti “Collegamento ad Angolo” e “Collegamento Centrale” della fase di test del “modello esteso”

4.6.1 Analisi dei risultati

Per analizzare i risultati del modello base, del modello aumentato e del modello esteso andiamo ad analizzare le traiettorie, densità, velocità e tempi all'interno degli ambienti proposti.

Validazione del modello base

Per valutare il modello base, sono stati utilizzati gli ambienti “Doppia Porta Opposta” e “Ancora”. Il nostro obiettivo è quello di mantenere gli stessi comportamenti ottenuti prima dell'introduzione del curriculum ad alta densità. In Figura 4.18 viene mostrato un confronto tra le traiettorie di “Doppia Porta Opposta” prima e dopo l'introduzione del curriculum ad alta densità. Possiamo osservare come, non solo abbiamo mantenuto i comportamenti, ma abbiamo una migliore compattezza all'interno del flusso e una maggiore distanza sociale tra i due gruppi.

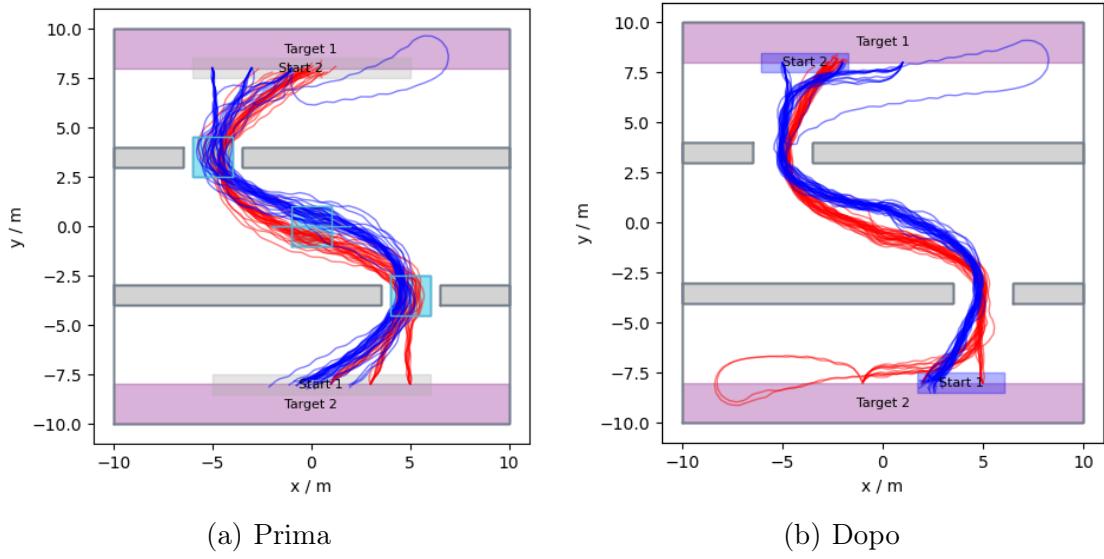


Figura 4.18: Confronto tra le traiettorie in “Doppia Porta Opposta” prima e dopo l’introduzione del curriculum ad alta densità

Validazione del modello aumentato

Per valutare il modello aumentato, sono stati utilizzati gli ambienti “Doppia Porta Opposta 7” e “Ancora 8”. Il nostro obiettivo è quello di apprendere i nuovi comportamenti per gestire correttamente scenari con elevato traffico. In Figura 4.19 viene mostrato un confronto tra le traiettorie di “Ancora 8” prima e dopo l’introduzione del curriculum ad alta densità. Possiamo osservare come le traiettorie sono più pulite e presentano meno valori anomali rispetto all’esecuzione senza curriculum ad alta densità.

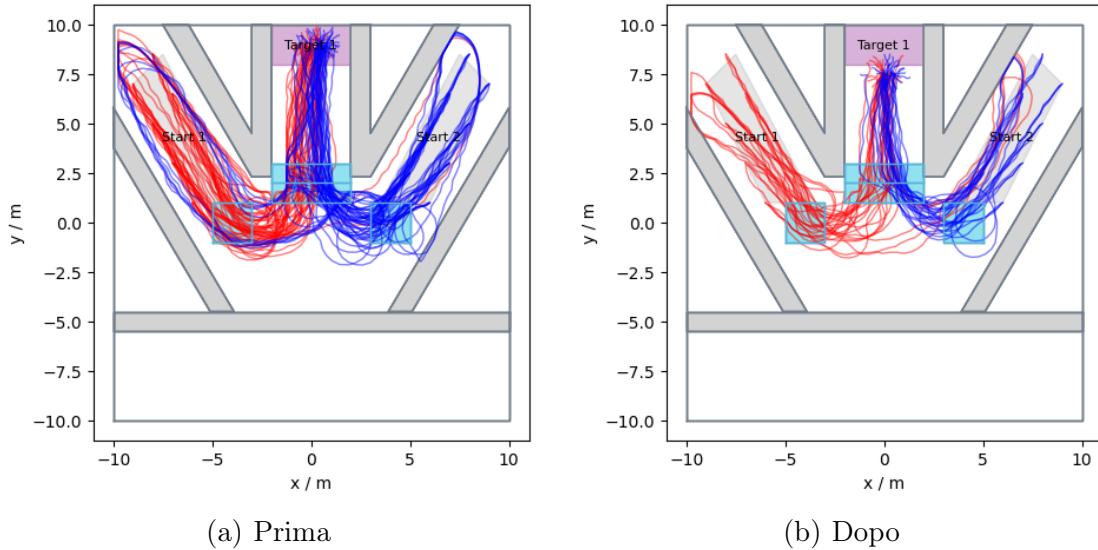


Figura 4.19: Confronto tra le traiettorie in “Ancora 8” prima e dopo l’introduzione del curriculum ad alta densità

Analisi della larghezza del corridoio prima di un bottleneck

Dopo aver verificato che i pedoni mantenevano i comportamenti osservati in precedenza e avevano appreso le norme comportamentali appropriate per affrontare situazioni di traffico intenso, si è proceduto a testare se i pedoni addestrati tramite reinforcement learning presentassero comportamenti analoghi a quelli dei pedoni umani.

A tale scopo, ci si aspetta che nei diversi ambienti “Imbuto” i risultati mostrino una maggiore densità e un traffico più intenso nei corridoi più ampi prima del punto di strozzatura, mentre si prevede un miglior flusso e tempi di evacuazione più rapidi nei corridoi più stretti.

La visualizzazione delle traiettorie fornisce un’analisi dettagliata di come i pedoni abbiano affrontato l’ambiente dal punto di vista comportamentale. In Figura 4.20 sono riportate le traiettorie nelle 3 configurazioni dell’ambiente. I pedoni entrano nel corridoio che precede il punto di strozzatura e, progressivamente, attraversano tale punto fino a completare il percorso all’interno dell’ambiente.

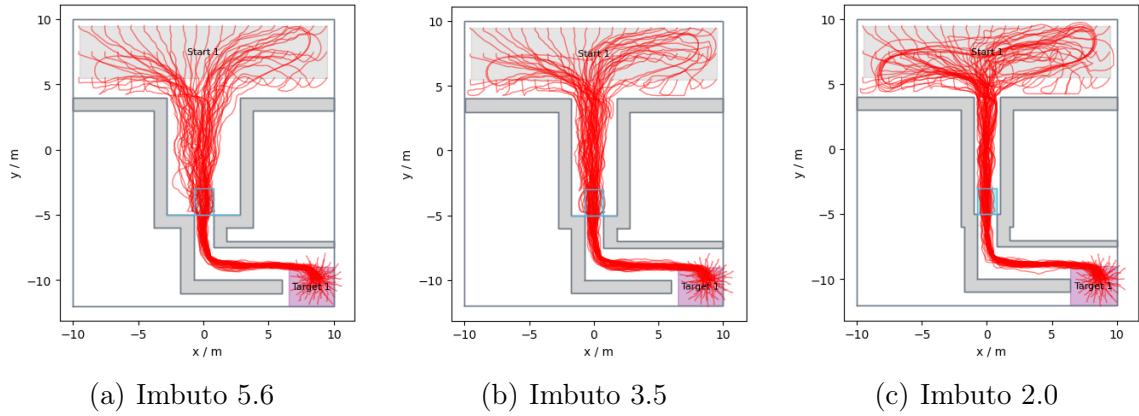


Figura 4.20: Traiettorie all'interno degli ambienti Imbuto

Per una migliore comprensione dell'accumulo dei pedoni all'interno del bottleneck, è possibile utilizzare la densità di Voronoi. Nella Figura 4.21, viene presentata la densità di Voronoi nell'ambiente caratterizzato dal corridoio più ampio. Tramite questa visualizzazione, riusciamo a notare come un lato di pedoni ha preso la precedenza mentre l'altro ha aspettato. È importante notare che una visualizzazione analoga è ottenuta anche nelle altre due versioni dell'ambiente.

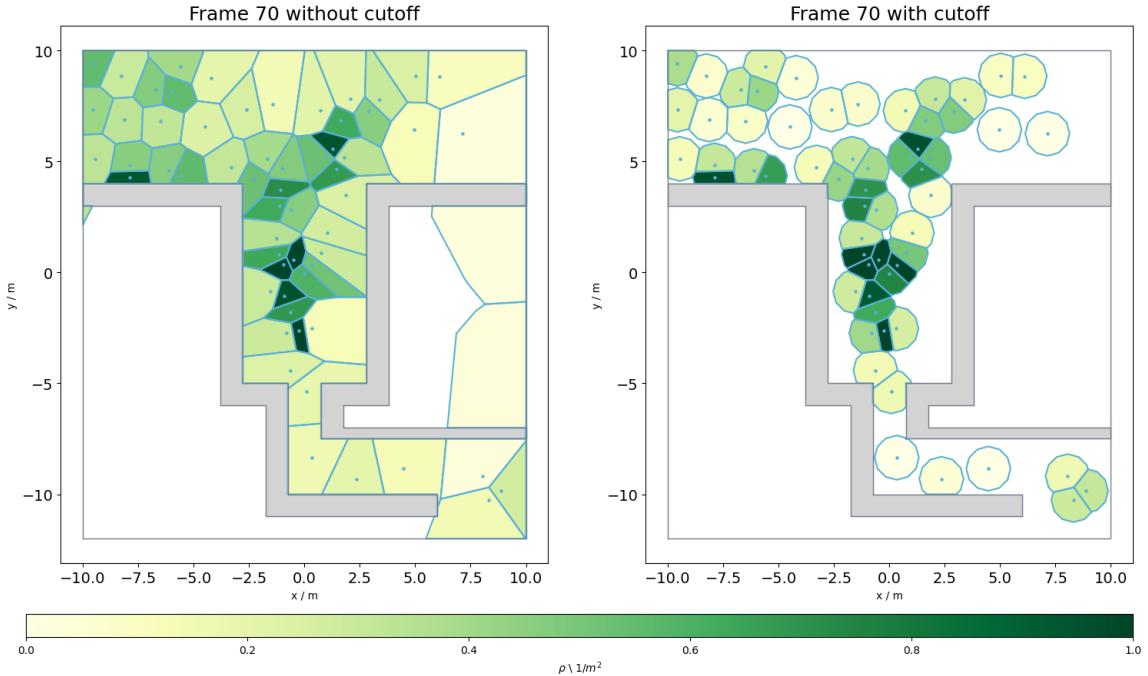


Figura 4.21: Punti di accumulo nell'ambiente Imbuto 5.6

Una visualizzazione che permette di individuare differenze tra le tre configurazioni dell'ambiente è la densità individuale. In questa visualizzazione, è stata posizionata una linea di misurazione nel punto in cui inizia la strozzatura. Da qui viene creata una seconda linea di misurazione virtuale a una distanza di 1 metro. Man mano che gli agenti attraversano il bottleneck

dalla prima linea alla seconda, viene calcolata la densità. Maggiore il tempo necessario all'agente per attraversare la sezione, maggiore la densità.

Questa visualizzazione fornisce due misure: in primo luogo, fornisce informazioni sulla quantità di densità presente, espressa dal numero di persone per metro quadro. In secondo luogo, la simmetria del plot fornisce indicazioni sul comportamento dei pedoni durante il passaggio: se il plot risulta simmetrico, significa che i pedoni hanno mantenuto le norme comportamentali corrette e che il flusso di pedoni è rimasto costante durante l'esecuzione.

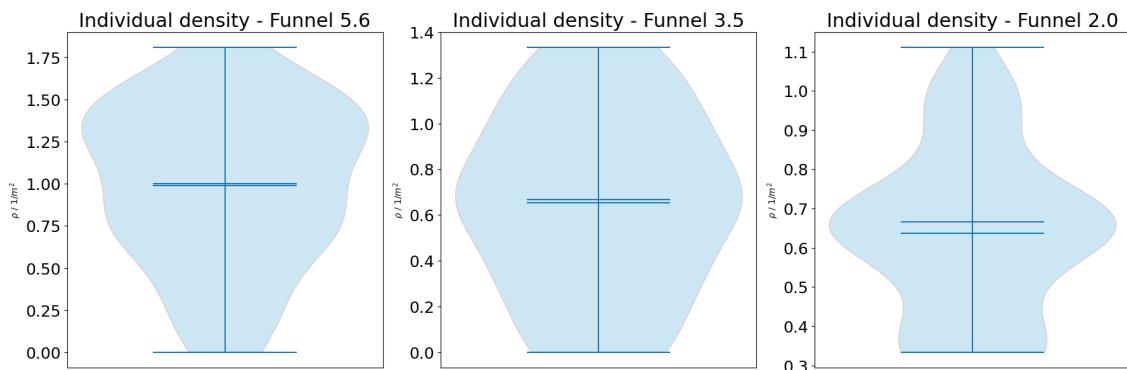


Figura 4.22: Densità individuale nelle 3 configurazioni di Imbuto

Viene riportata una breve analisi della Figura 4.22:

- **Imbuto 5.6:**

- La distribuzione della densità varia tra 0 e circa 1.75 persone al metro quadrato.
- La mediana è approssimativamente a 1.0 persona al metro quadrato.
- La forma del grafico mostra una distribuzione leggermente bimodale con due picchi di densità intorno a 0.75 e 1.35

- **Imbuto 3.5:**

- La distribuzione della densità varia tra 0 e circa 1.4 persone al metro quadrato.
- La mediana è attorno a 0.65 persone al metro quadrato.
- La forma del grafico indica una distribuzione simmetrica nella sua interezza

- **Imbuto 2.0:**

- La distribuzione della densità varia tra 0.35 e circa 1.1 persone al metro quadrato.
- La mediana è attorno a 0.65 persone al metro quadrato.
- La forma del grafico a violino mostra una distribuzione abbastanza simmetrica, con un grande picco intorno alla media.

In generale, si nota che la densità individuale tende a diminuire con la riduzione della dimensione del corridoio prima della strozzatura, e che i due ambienti con il corridoio più stretto hanno una simmetria migliore rispetto a quello con un corridoio largo.

Per verificare che l’ambiente con il corridoio più ampio rappresenti la versione peggiore tra le tre configurazioni del collo di bottiglia, è stato effettuato un confronto delle densità di Voronoi mediante un’area di misurazione situata 1 metro prima del passaggio. Questa analisi ci consente di osservare in dettaglio come la distribuzione spaziale delle persone vari in funzione della larghezza del corridoio, fornendo una chiara rappresentazione delle densità relative e delle dinamiche di flusso in ciascuna configurazione.

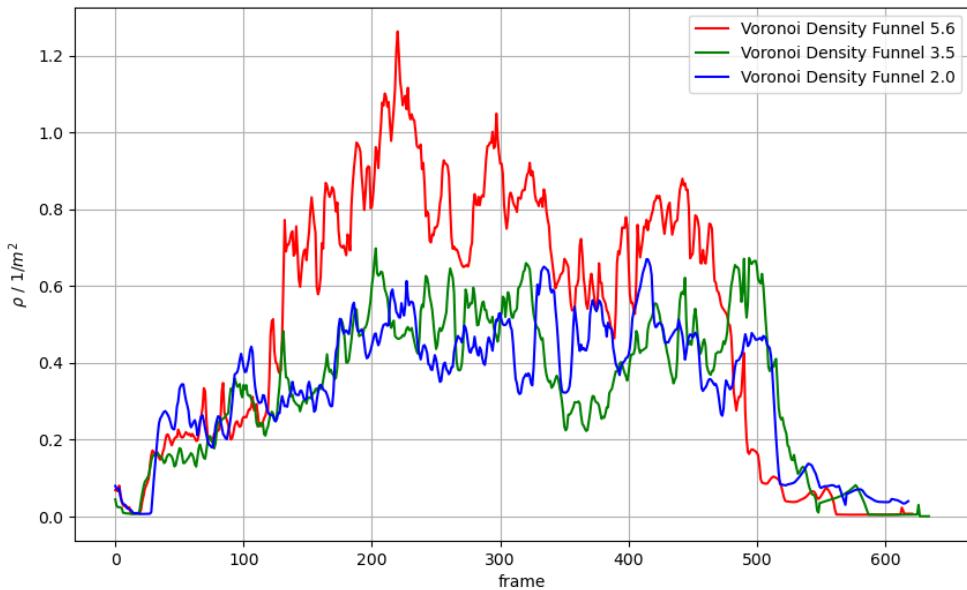


Figura 4.23: Densità di Voronoi nelle 3 configurazioni di Imbuto

Come è facile osservare, durante quasi l’intera esecuzione, 5.6 presenta una densità maggiore rispetto alle altre due versioni, le quali si alternano nel rappresentare l’ambiente più favorevole. Questo risultato è in accordo con quanto affermato nello studio: all’aumentare della larghezza del corridoio davanti a una strozzatura, si osserva un incremento della densità e una conseguente creazione di maggiore traffico. I nostri pedoni, addestrati tramite reinforcement learning, hanno quindi mantenuto un comportamento simile

a quello di pedoni umani, dimostrando la validità del modello simulativo utilizzato per l'analisi.

Confronto dei tempi di evacuazione di collegamenti centrali e ad angolo

Per questo studio sono state posizionate una linea di misurazione e tre aree di misurazione per ciascun ambiente: due aree sono collocate al collegamento tra le stanze laterali e la stanza centrale, mentre l'area rimanente è situata a 1 metro dal passaggio finale. Ciò che ci aspettiamo dall'ambiente, in linea con quanto emerso dallo studio, è che l'ambiente con il collegamento ad angolo mostri un tempo di evacuazione locale, e quindi della stanza, migliore rispetto ai collegamenti tra stanze centrali. Tuttavia, ci si attende anche che il tempo di evacuazione globale, e quindi dell'intero ambiente, sia peggiore rispetto all'altra versione.

Anche in questo caso, viene fornita la visualizzazione delle traiettorie nei due ambienti. Sebbene questa visualizzazione non ci aiuti direttamente a comprendere i tempi e le densità dell'ambiente, è utile per osservare, oltre alla posizione delle varie aree di misurazione, le dinamiche di flusso nei diversi ambienti. Come si può osservare, i pedoni non hanno mantenuto delle traiettorie pulite, creando dei giri più larghi per affrontare il corridoio centrale.

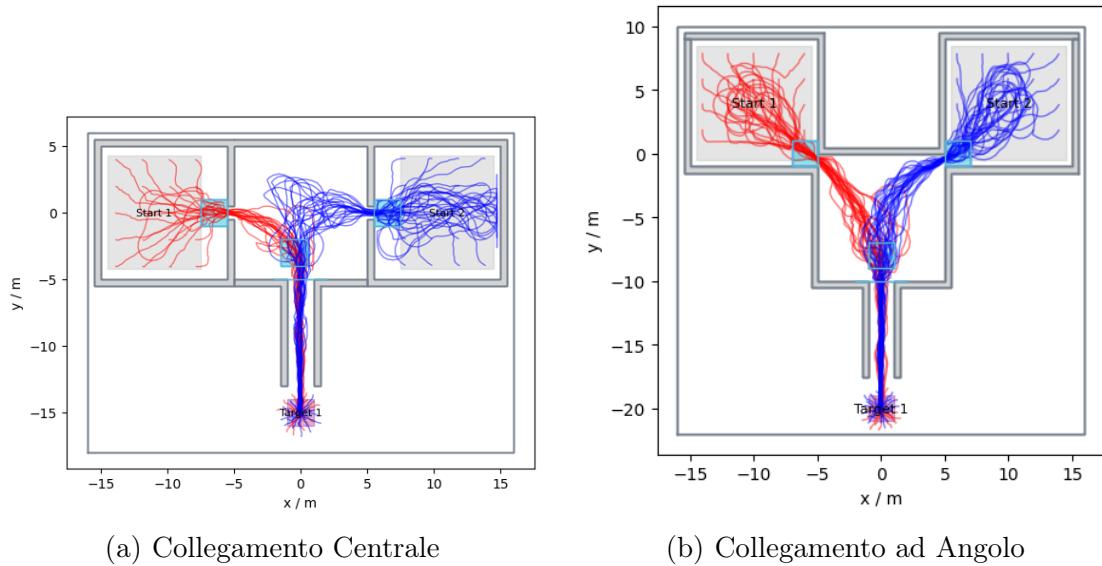


Figura 4.24: Traiettorie all'interno degli ambienti "Collegamento Centrale" e "Collegamento ad Angolo"

Per visualizzare i punti di accumulo dei pedoni utilizziamo la densità di Voronoi. Come è possibile osservare dalla Figura 4.25 e 4.26, i due gruppi di

pedoni sono riusciti a coordinarsi con successo, raggiungendo in pochi punti densità di una persona per metro quadrato.

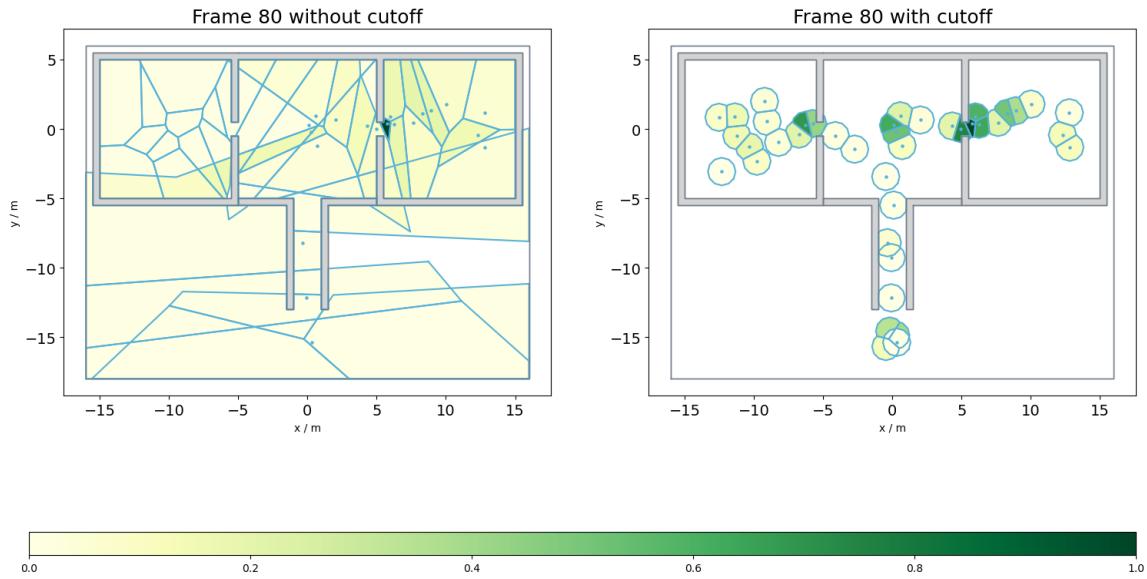


Figura 4.25: Densità: punti di accumulo nel Collegamento Centrale

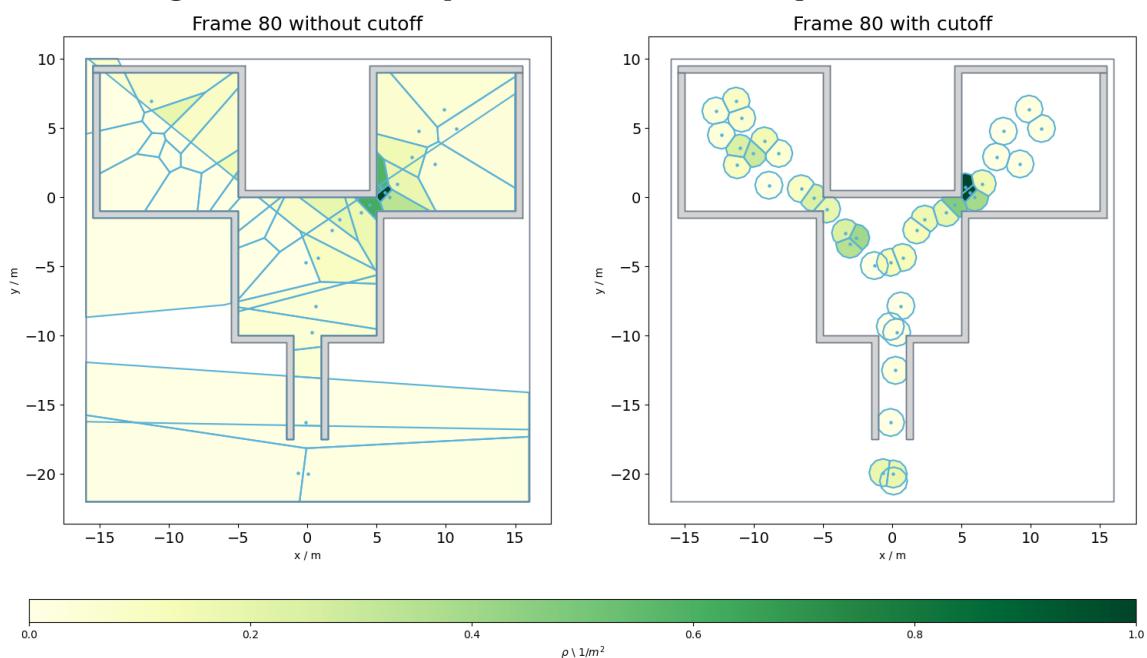


Figura 4.26: Densità: punti di accumulo nel Collegamento ad Angolo

Per analizzare i tempi di evacuazione dei due ambienti possiamo utilizzare un confronto di densità dei due ambienti, in quanto mostrano l’andamento del numero di pedoni per metro quadrato all’avanzamento del tempo.

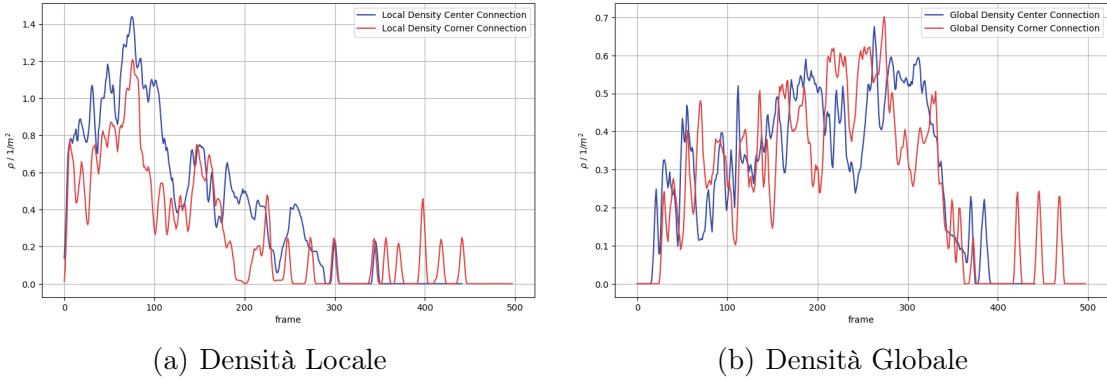


Figura 4.27: Confronto di densità locale e globale negli ambienti "Collegamento Centrale" e "Collegamento ad Angolo"

Come è possibile osservare dalla Figura 4.27(a) l’ambiente con i collegamenti agli angoli ha densità minore per tutta la durata dell’esecuzione, in accordo con quanto emerso dagli studi eseguiti. Per quanto riguarda la Figura 4.27(b), riusciamo ad osservare come, effettivamente, l’ambiente con i collegamenti al centro termina prima rispetto alla sua controparte, in accordo con quanto emerge dallo studio.

In conclusione possiamo affermare che i nostri pedoni, allenati tramite reinforcement learning, hanno mantenuto, anche per gli ambienti con connessioni, dei comportamenti simili a quelli umani.

4.7 Analisi di Sensitività

4.7.1 Introduzione

In parallelo all’estensione del modello è stata condotta un’analisi di sensitività con l’obiettivo di ottimizzare i tempi di addestramento, mantenendo comunque le prestazioni del modello illustrate nei capitoli precedenti. È stato inoltre realizzato uno studio ablativo per identificare il ruolo e l’importanza delle varie componenti nel determinare le prestazioni del modello.

Da questo studio è emerso che, modificando la rete neurale - che inizialmente era quella di default di Stable Baselines 3 - la fase di retraining non era più necessaria, permettendo una riduzione dei tempi di addestramento di circa 2/3, passando da 35/40 minuti a circa 15/20 minuti. È stato possibile constatare ciò poiché, dopo aver eseguito la fase di training senza la fase di retraining e con le modifiche alla rete, la fase di testing non solo ha mantenuto gli stessi comportamenti, ma ha mostrato miglioramenti in alcuni casi.

In Figura 4.28 è possibile osservare la durata dell'esecuzione ottenuta nello studio parallelo.

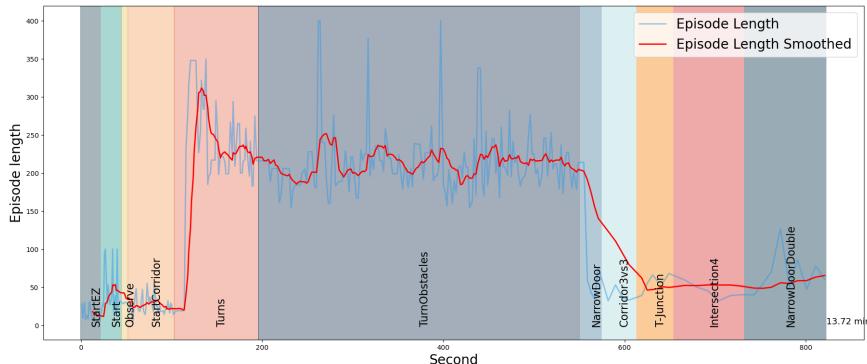


Figura 4.28: Lunghezza Episodi

Si è quindi deciso, dopo aver terminato con successo il modello esteso mostrato nella sezione precedente, di analizzare se tale rete fosse utilizzabile anche in un modello ad alta densità, considerato il grande risparmio di tempo ottenuto. È stata eseguita una sessione di training con il curriculum per l'alta densità, utilizzando la rete che aveva offerto le migliori prestazioni nel modello base. Il training è avvenuto con successo, terminando in circa 17 minuti. Tuttavia, durante la fase di test, gli ambienti più complessi del modello ad alto traffico, come i vari "Imbuto" o le "Connesioni", non riuscivano a progredire, terminando in minimi locali.

Questo fenomeno è stato causato dal fatto che la rete utilizzata era composta da 3 layer nascosti con 128 neuroni ciascuno, mentre le feature da considerare all'interno del progetto, soprattutto in presenza di molti altri pedoni, erano in numero maggiore. Gli ambienti con più pedoni, seppur utilizzando un algoritmo a singolo agente, presentavano una notevole stocasticità. Ciò rendeva difficile discernere se la nuova decisione da prendere fosse relativa al comportamento del singolo agente o degli agenti adiacenti. Questo, insieme al fatto che le osservazioni per singolo agente erano 185, numero maggiore di quello di ciascun layer della rete, ha causato il malfunzionamento nel modello ad alta densità.

Si è dunque deciso di provare diverse configurazioni per verificare se fosse possibile addestrare gli agenti a mantenere comportamenti adeguati in scenari ad alta densità con sessioni di training che non presentano la fase di retraining, permettendo dunque di ottenere un curriculum learning senza estensioni.

4.7.2 Configurazioni

Rete 256-256-128

La prima configurazione utilizzata presenta un raddoppio dell'ampiezza dei primi due strati nascosti della rete, passando da 128 a 256 neuroni. L'andamento dell'addestramento è visualizzabile in Figura 4.29

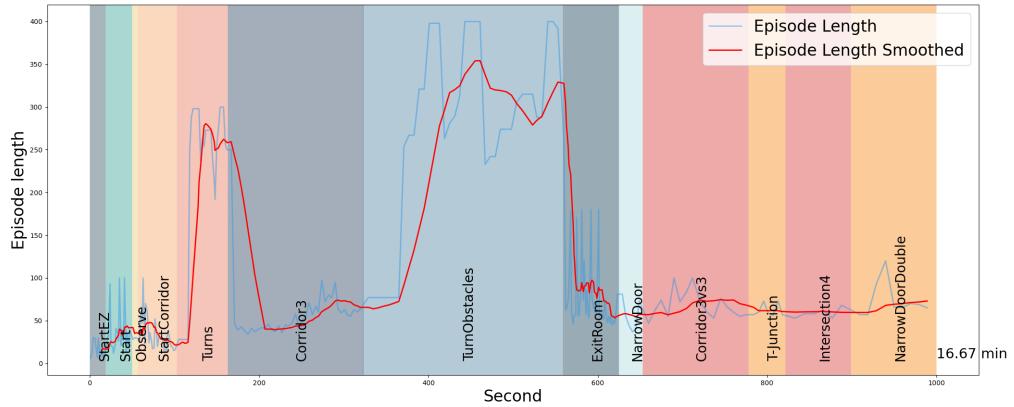


Figura 4.29: Durata in secondi degli ambienti del curriculum: Rete 256-256-128

Rete 256-256-128 con Reward collisioni

Successivamente, è stata mantenuta la stessa configurazione di rete ma sono stati riabilitati i reward negativi per la prossemica a lunga e media gittata, al fine di valutare se con una modifica alla rete si potesse ritornare al modello base di reward. L'andamento dell'addestramento è visualizzabile in Figura 4.30.

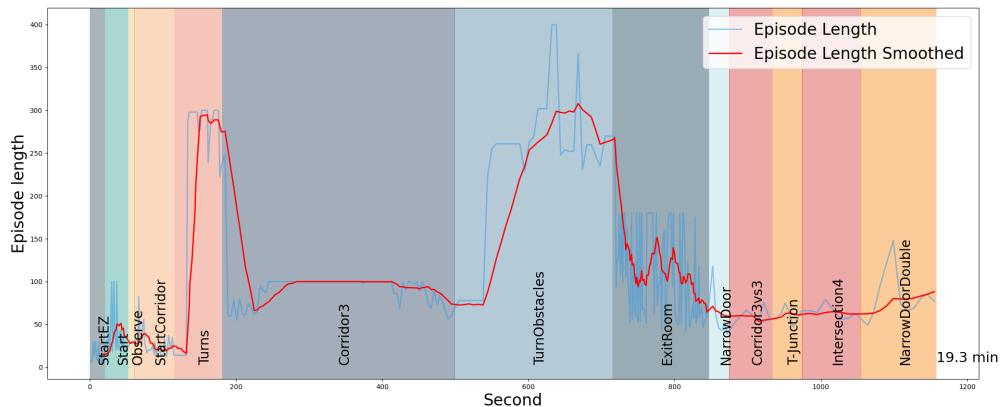


Figura 4.30: Durata degli ambienti del curriculum: Rete 256-256-128 con Reward collisioni

Rete 256-128-64

La terza configurazione va a riprendere, dal punto di vista dei reward per le collisioni, la prima, ma modifica la rete neurale, in quanto abbiamo 256 neuroni nel primo strato nascosto, 128 nel secondo e solo 64 nel terzo. L’andamento dell’addestramento è visualizzabile in Figura 4.31

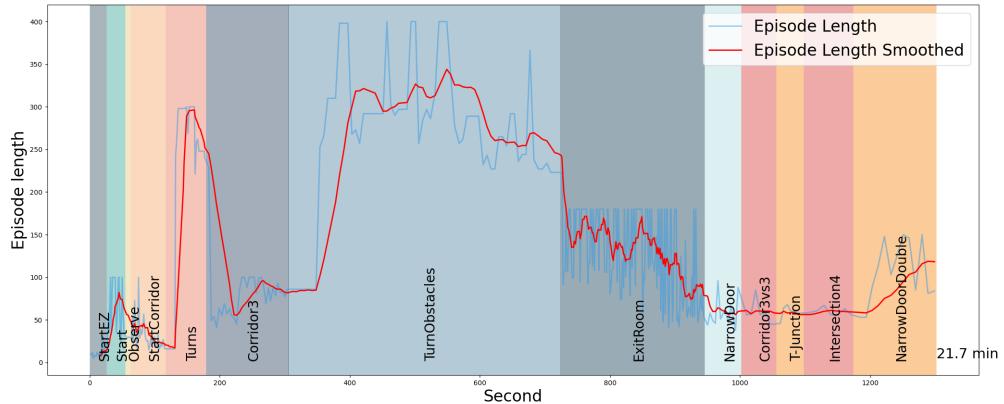


Figura 4.31: Durata degli ambienti del curriculum: Rete 256-128-64

Rete 256-128-64 con Reward collisioni

Infine, per concludere i test, sono state combinate la nuova rete con la riabilitazione dei reward per le collisioni. L’andamento dell’addestramento è visualizzabile in Figura 4.32

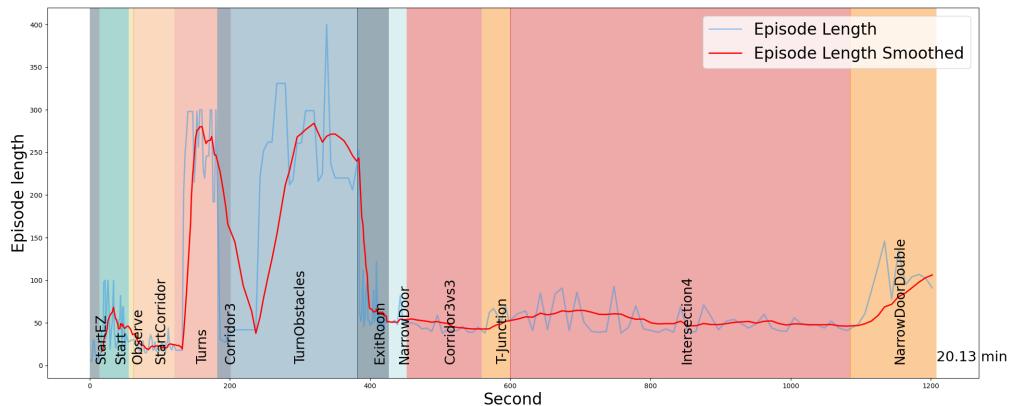


Figura 4.32: Durata degli ambienti del curriculum: Rete 256-128-64 con Reward collisioni

4.7.3 Risultati

Per testare l’efficacia dell’addestramento, è stata utilizzata per tutte le configurazioni una struttura simile a quella adottata per il modello esteso, comprendente tre fasi principali:

1. **Modello base:** Si è verificato che vengano mantenuti i comportamenti del modello base tramite i livelli “Ancora” e “Doppia Porta Opposta”.
2. **Modello aumentato:** È stato valutato se i pedoni hanno imparato a gestire adeguatamente scenari ad alto traffico e con interazioni sociali complesse, utilizzando ambienti come “Ancora 8” e “Doppia Porta Opposta 7”.
3. **Modello esteso:** È stato valutato se i pedoni allenati tramite reinforcement learning mantengono comportamenti simili a quelli umani in casi di test specifici, utilizzando le diverse configurazioni degli ambienti “Imbuti” e “Collegamenti”.

Dall’analisi delle fasi di test emerge che tutte e quattro le configurazioni eseguite hanno mostrato prestazioni relativamente positive nelle prime due fasi. Di conseguenza, ci concentreremo sull’analisi del comportamento nella terza fase. Verranno mostrate, di seguito, le configurazioni, ordinate da quella con le peggiori performance a quella con le migliori.

Risultati Rete 256-256-128

In tutti gli scenari proposti durante la fase di test del modello esteso, questa configurazione ha portato a degli stalli improvvisi nei pedoni. Delle quattro configurazioni si è rivelata, senza dubbio, la peggiore.

Risultati Rete 256-128-64 con Reward Collisioni

Come la configurazione precedentemente testata, è stato ottenuto lo stesso risultato ma con un comportamento diverso: se precedentemente gli agenti avevano stalli improvviso, adesso vanno velocemente verso il centro della stanza e iniziano a girare, creando uno stallo. Per questo motivo si contende il posto di peggior configurazione testata, insieme al primo modello analizzato.

Risultati Rete 256-128-64

In questa configurazione, gli agenti, all’interno degli ambienti “Collegamenti”, si bloccavano ai passaggi in quanto non avevano sviluppato un buon coordinamento con l’altro gruppo, e nessuno dei due voleva intraprendere la stanza principale. Per quanto concerne gli ambienti “Imbuto”, invece, essendo ambienti a singolo gruppo, abbiamo ottenuto dei buoni risultati, addirittura migliori rispetto al modello base da un punto di vista delle traiettorie e della quantità di densità creata.

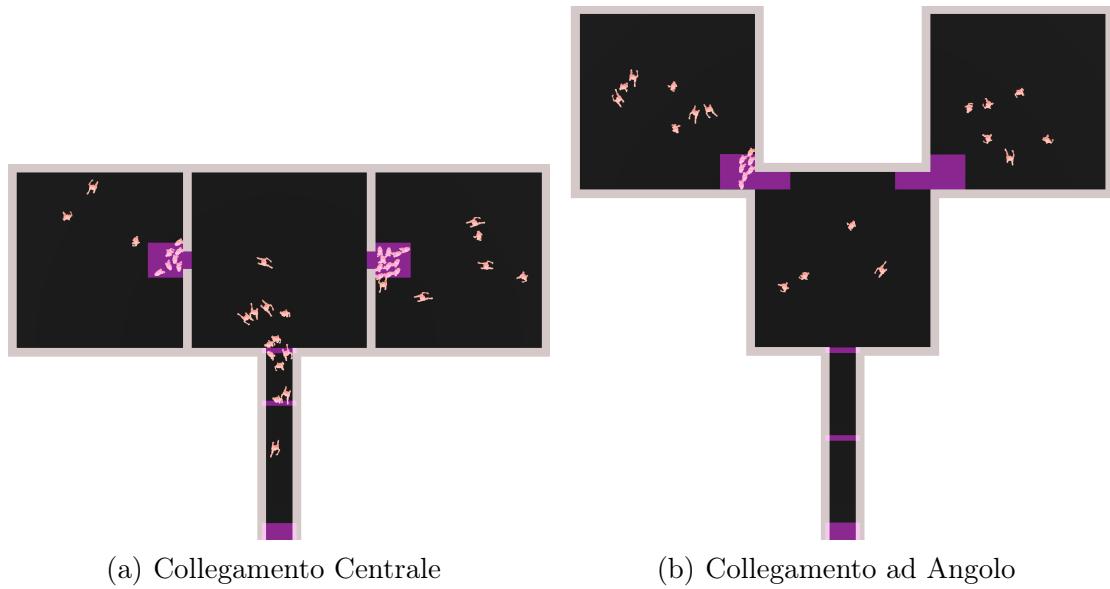


Figura 4.33: Stalli nei Collegamenti a causa della scarsa coordinazione tra agenti

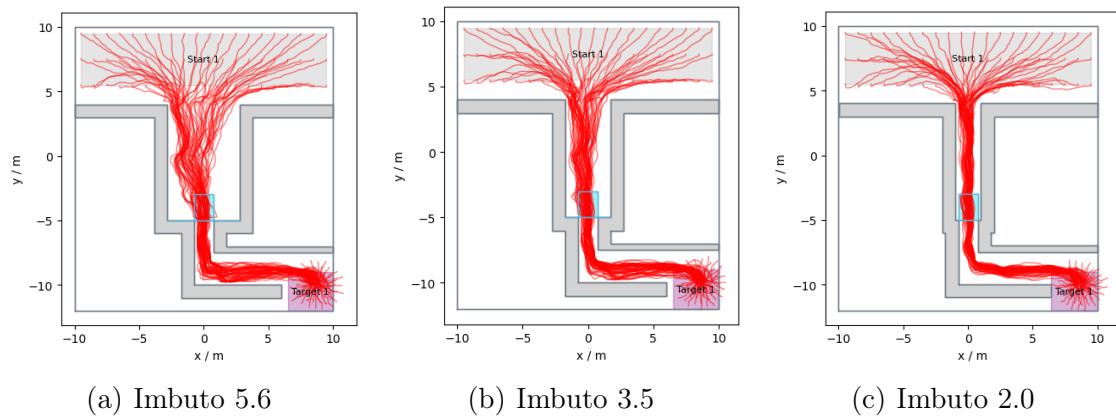


Figura 4.34: Traiettorie degli ambienti "Imbuto 5.6", "Imbuto 3.5" e "Imbuto 2.0" con rete aumentata.

Nonostante queste buone metriche, i risultati ottenuti non riflettono il comportamento aspettato dagli studi, in quanto l'ambiente con il corridoio più largo è il primo a terminare, nonostante abbia densità peggiori delle altre due versioni.

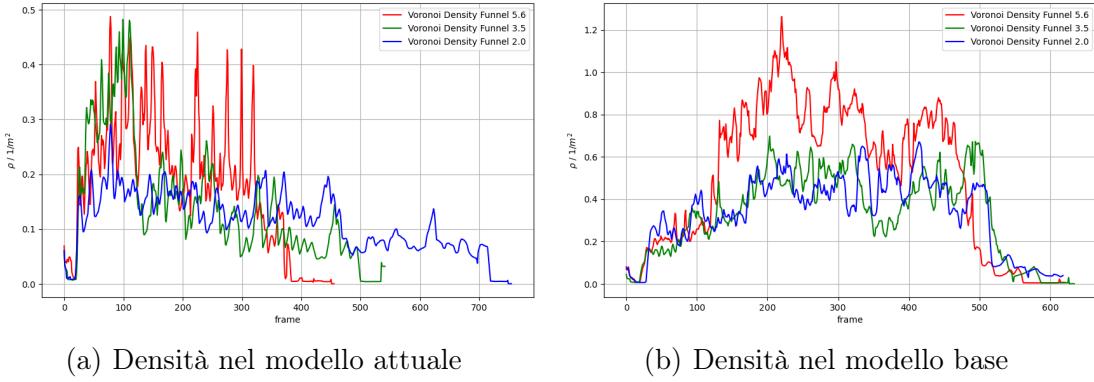


Figura 4.35: Confronto della densità tra modello base e modello con rete aumentata all'interno degli ambienti "Imbuto"

Risultati Rete 256-256-128 con Reward Collisioni

La seguente configurazione è quella che ha ottenuto i risultati migliori. Analizzando gli scenari, emergono i seguenti comportamenti:

Collegamento Centrale L'ambiente è stato completato con successo, sebbene con un comportamento diverso rispetto al modello base. Invece di coordinarsi per il passaggio all'interno della stanza centrale, il gruppo di destra ha dato precedenza a tutto il gruppo di sinistra prima di proseguire con il proprio percorso. Questo comportamento, nonostante offra buone traiettorie e densità, non rispecchia un comportamento umano.

Collegamento ad Angolo Anche in questo ambiente, come nella sua controparte, un gruppo ha ricevuto la precedenza sull'altro. Tuttavia, questo comportamento è stato meno marcato, probabilmente a causa della scarsa visibilità della presenza dell'altro gruppo.

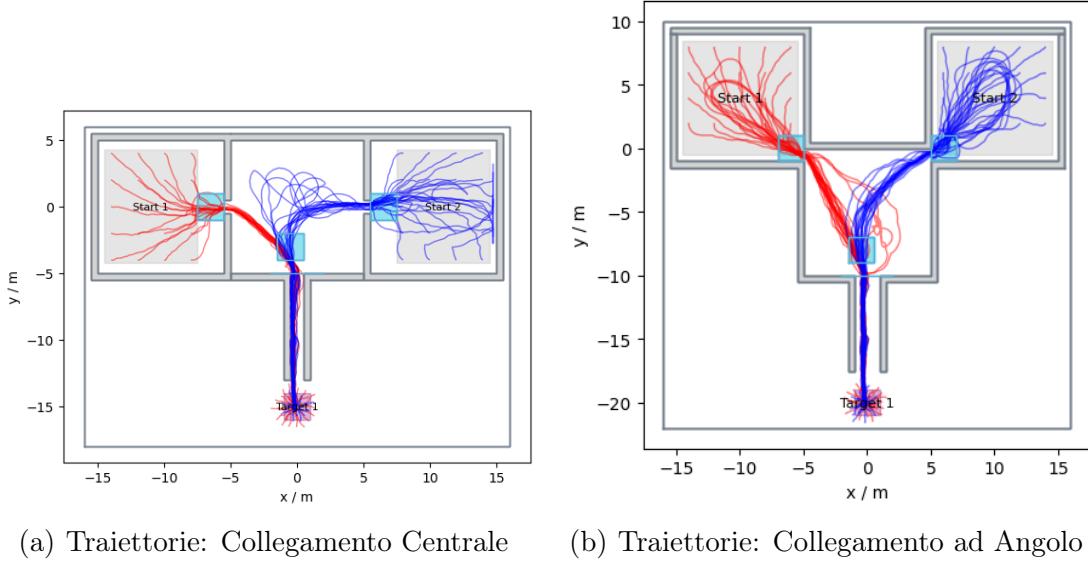


Figura 4.36: Un gruppo di pedoni dà la precedenza all’altro

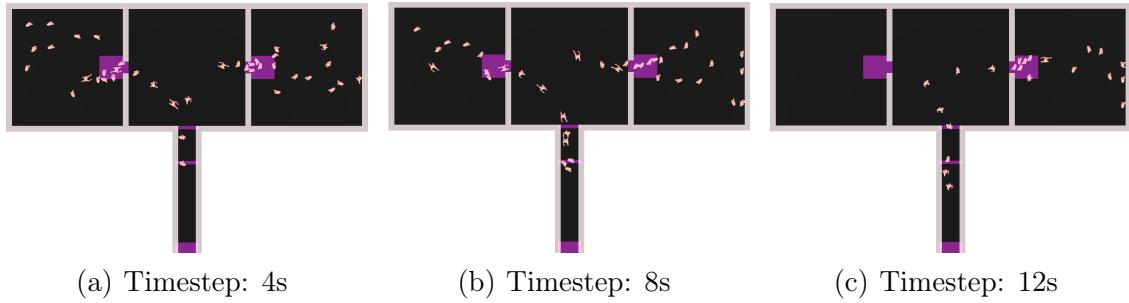


Figura 4.37: Visualizzazione del flusso non uniforme tra le due stanze del Collegamento Centrale

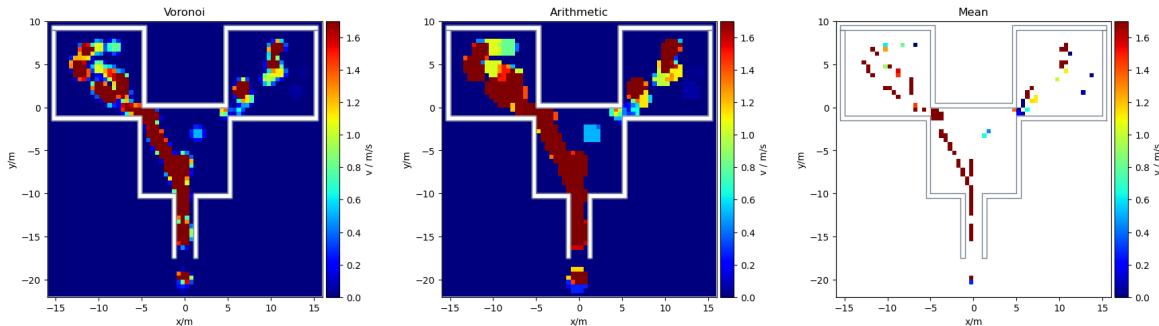


Figura 4.38: Il flusso di sinistra prosegue ad alta velocità, mentre quello di destra aspetta

Imbuto Nelle varie configurazioni di questo ambiente, si sono ottenuti risultati migliori rispetto al modello con retraining. Questo è dovuto al fatto che, invece di avere un lato di agenti che intraprendeva il corridoio, come avveniva nel modello con retraining, i pedoni più vicini al corridoio hanno avanzato progressivamente lungo l’ambiente. Questo elemento ci ha permesso non solo

di ottenere dati qualitativi migliori rispetto al modello base, come avveniva nel modello precedente, ma anche di avere un analisi realistica e permetterci di avere pedoni con comportamenti simili a quelli umani.

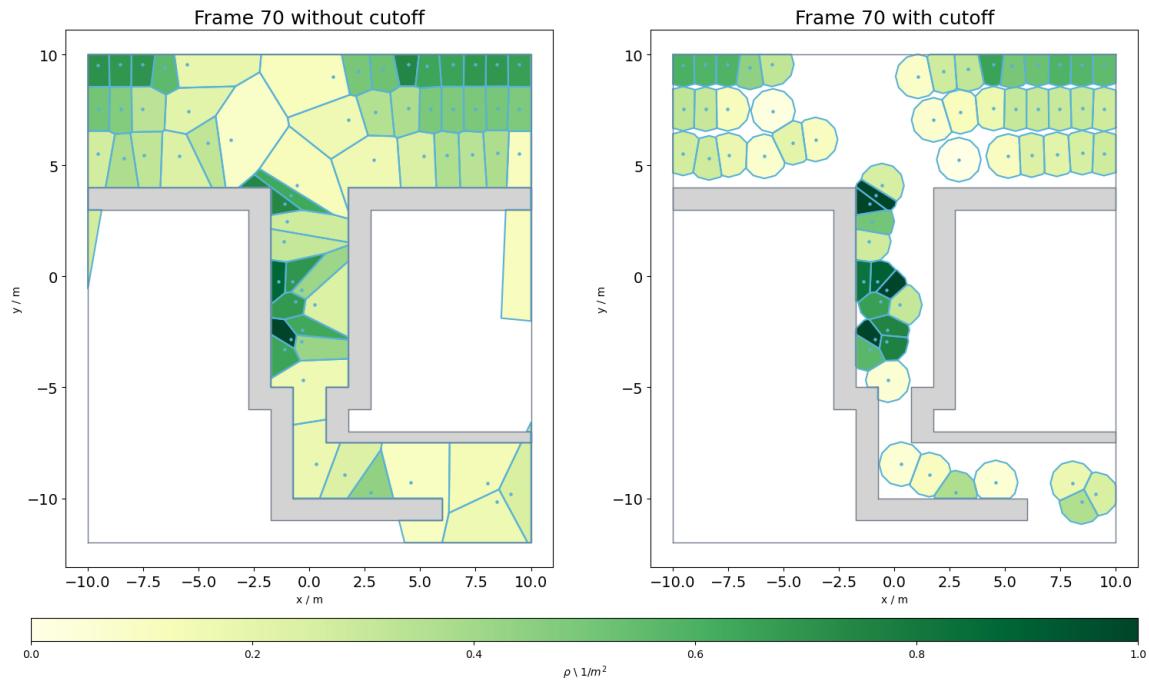


Figura 4.39: Visualizzazione dei punti di accumulo dei pedoni all'interno di Imbuto

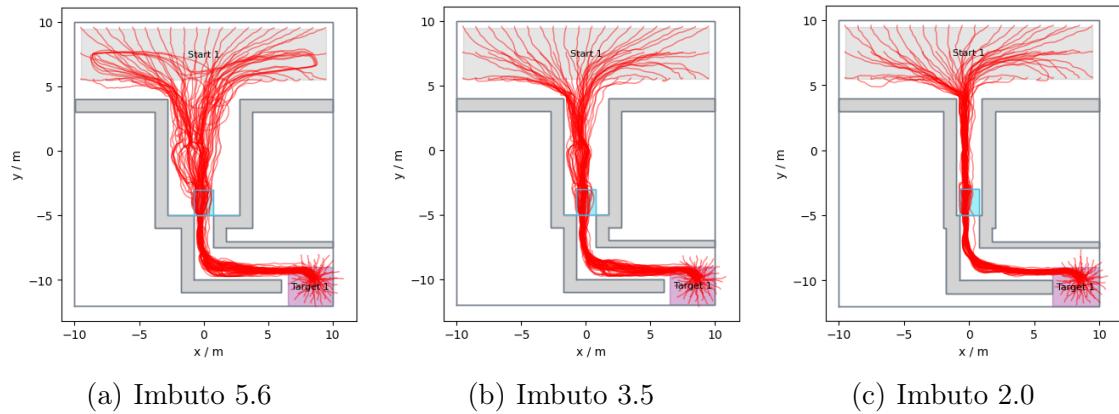
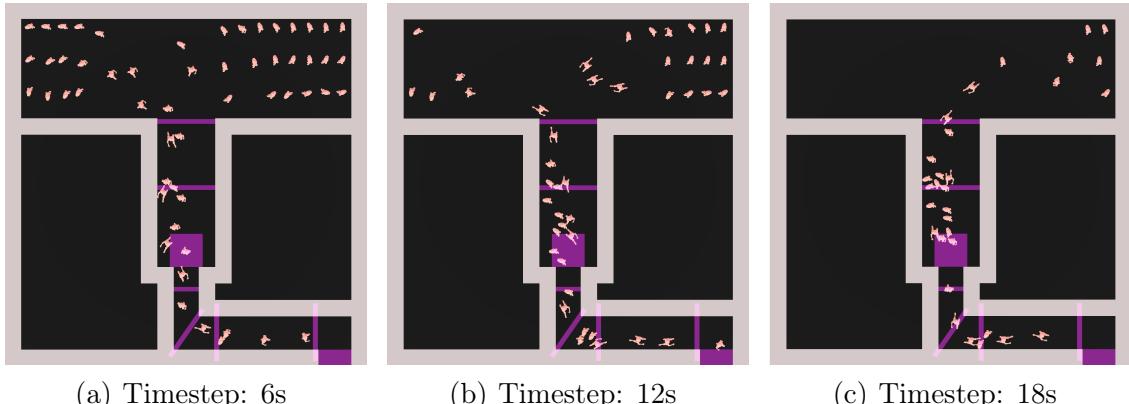


Figura 4.40: Traiettorie degli ambienti "Imbuto 5.6", "Imbuto 3.5" e "Imbuto 2.0" con rete aumentata e con reward negativi per collisioni.

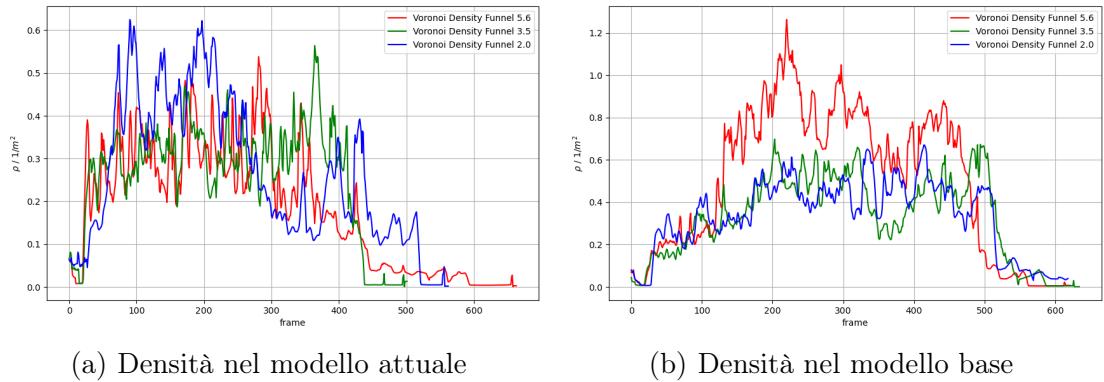


(a) Timestep: 6s

(b) Timestep: 12s

(c) Timestep: 18s

Figura 4.41: Esecuzione dell'ambiente "Imbuto 3.5"



(a) Densità nel modello attuale

(b) Densità nel modello base

Figura 4.42: Confronto della densità tra modello base e modello con rete aumentata e con reward negativi per collisioni all'interno degli ambienti "Imbuto"

Capitolo 5

Conclusione e sviluppi futuri

Questa tesi si è incentrata sulla realizzazione di agenti intelligenti utilizzando il Reinforcement Learning e sulla creazione di un sistema per la simulazione pedonale in cui allenarli e testarli.

Durante la prima parte del progetto, è stato sviluppato un sistema per la gestione completa delle simulazioni pedonali, offrendo la possibilità di creare nuovi ambienti e agenti. Il sistema è completamente e facilmente personalizzabile, permettendo di essere utilizzato sia per l’addestramento degli agenti sia per testare differenti modelli di simulazione pedonale.

Tuttavia, il modello iniziale non era adeguato per rappresentare comportamenti di folle e scenari ad alta densità. Per affrontare questa limitazione, nella seconda parte del progetto è stato esteso il modello originale. Questa fase ha comportato modifiche agli ambienti preesistenti e la creazione di nuovi, in modo da fornire ai pedoni una conoscenza sufficiente degli ambienti caratterizzati da elevato traffico.

L’estensione del modello ha permesso di creare una simulazione capace di apprendere e gestire situazioni ad alta densità. Per verificare che i comportamenti dei pedoni simulati fossero realistici e utilizzabili in simulazioni di folle reali, sono stati considerati due casi di studio. Le metriche ottenute dalle nostre simulazioni sono state confrontate con quelle presentate negli studi, evidenziando come i nostri pedoni, allenati tramite il reinforcement learning, siano riusciti a replicare i comportamenti dei pedoni reali descritti nei casi di studio.

Un’analisi di sensitività è stata eseguita sul modello ottenuto. Parallelamente, uno studio ha dimostrato che la fase di retraining in scenari a basso traffico, oltre a raddoppiare o triplicare i tempi di training, portava a prestazioni peggiori nella fase di test. Per verificare se tale risultato si potesse ottenere anche nel modello esteso, abbiamo condotto vari test senza retraining, utilizzando scenari ad alto traffico. Nonostante alcune configurazioni abbiano mostrato buone esecuzioni in termini di densità e tempi di esecuzio-

ne, nessuna è riuscita a replicare fedelmente i comportamenti attesi dai casi di studio analizzati, a causa della forte stocasticità legata all'elevato numero di pedoni presenti negli ambienti.

Questi risultati evidenziano la complessità e le sfide legate all'addestramento di modelli in situazioni ad alto traffico pedonale tramite un algoritmo a singolo agente come PPO. La ricerca futura dovrebbe concentrarsi sull'ottimizzazione di tali modelli e sull'esplorazione di approcci multi-agente che potrebbero meglio catturare le dinamiche delle folle in scenari ad alta densità.

Il progetto descritto in questa relazione si distingue per la sua elevata personalizzabilità e facilità di modifica, aprendo la strada a numerosi possibili sviluppi futuri.

Un primo sviluppo possibile è l'introduzione della completa eterogeneità dei pedoni. In questo scenario, ogni pedone potrebbe essere dotato di un comportamento peculiare che non riguarda solo la velocità desiderata, ma anche la distanza che desidera mantenere dagli altri pedoni, la tendenza a stare vicino ai muri, e così via. La varietà di comportamenti individuali modellabili è estremamente ampia, il che potrebbe portare a simulazioni più imprevedibili e dinamiche. Questi miglioramenti non solo renderebbero le simulazioni più realistiche, ma permetterebbero anche di studiare una gamma più vasta di scenari, migliorando la nostra comprensione dei comportamenti pedonali in situazioni diverse.

Inoltre, come già anticipato, un ulteriore sviluppo futuro potrebbe essere la creazione di un sistema con algoritmo di apprendimento multi-agente . Attualmente, il modello utilizzato è basato su un singolo agente, e questo significa che le osservazioni e i reward sono centrati su un pedone considerato individualmente e non sul comportamento del gruppo. Questo può costituire un problema, soprattutto in situazioni con molti pedoni, poiché all'aumentare del numero di pedoni aumenta anche la stocasticità dell'ambiente. Non è sempre possibile determinare se un comportamento specifico è il risultato dell'azione di un singolo pedone o dell'interazione con altri pedoni. Implementare un sistema con algoritmo di apprendimento multi-agente permetterebbe di modellare meglio queste interazioni, offrendo una rappresentazione più accurata e dinamica dei comportamenti collettivi all'interno delle simulazioni pedonali.

Uno sviluppo futuro del progetto può riguardare l’automazione dei vari processi. Al momento, vi sono diverse configurazioni e file necessari per eseguire il progetto, il che porta al beneficio di un’alta personalizzazione delle fasi di training e testing, ma al tempo stesso rischia di creare discrepanze all’interno dell’applicazione. Si potrebbe pensare di unificare alcuni di questi processi e implementare parser più efficienti per garantire un coordinamento ottimale tra i file. Questo migliorerebbe la coerenza del sistema, ridurrebbe il rischio di errori dovuti a configurazioni incoerenti e renderebbe il processo complessivo più snello e intuitivo, facilitando l’uso e la manutenzione del sistema.

Un ulteriore sviluppo del progetto può consistere nell’aggiunta di nuove zone o oggetti nell’ambiente simulato. Ad esempio, si potrebbero includere zone di attesa, zone in cui fare la fila, aree di maggiore interesse, oppure elementi come tornelli, macchinette per i biglietti, sedie, scale mobili, ascensori, ecc. Le possibilità di modifica sono infinite e possono permettere di modellare scenari più realistici, aumentando così la complessità e la validità delle simulazioni.

In conclusione, questi sviluppi futuri potrebbero significativamente migliorare la qualità e la versatilità del simulatore di pedoni e folle, rendendolo uno strumento ancora più potente per lo studio dei comportamenti pedonali e la progettazione di spazi pubblici efficienti.

Bibliografia

- [1] J. Adrian, A. Seyfried e A. Sieben. «Crowds in front of bottlenecks at entrances from the perspective of physics and social psychology». In: *Journal of The Royal Society Interface* 17.165 (apr. 2020) (cit. a p. 55).
- [2] Armin Seyfried Antoine Tordeux Mohcine Chraibi e Andreas Schadschneider. «Prediction of pedestrian dynamics in complex architectures with artificial neural networks». In: *Journal of Intelligent Transportation Systems* 24.6 (2020), pp. 556–568. DOI: 10.1080/15472450.2019.1621756. eprint: <https://doi.org/10.1080/15472450.2019.1621756>. URL: <https://doi.org/10.1080/15472450.2019.1621756> (cit. a p. 3).
- [3] Stefania Bandini, Sara Manzoni e Giuseppe Vizzari. «Agent Based Modeling and Simulation: An Informatics Perspective». In: *Journal of Artificial Societies and Social Simulation* 12.4 (2009), p. 4 (cit. a p. 3).
- [4] Stefania Bandini et al. «Distance-based affective states in cellular automata pedestrian simulation». In: *Nat. Comput.* 23.1 (2024), pp. 71–83. DOI: 10.1007/S11047-023-09957-Y. URL: <https://doi.org/10.1007/s11047-023-09957-y> (cit. a p. 3).
- [5] Yoshua Bengio et al. «Curriculum Learning». In: 2016, pp. 41–48. URL: <https://dl.acm.org/doi/10.1145/1553374.1553380> (cit. a p. 13).
- [6] C Burstedde et al. «Simulation of pedestrian dynamics using a two-dimensional cellular automaton». In: *Physica A: Statistical Mechanics and its Applications* 295.3 (giu. 2001). ISSN: 03784371. DOI: 10.1016/S0378-4371(01)00141-8. URL: <https://www.sciencedirect.com/science/article/pii/S0378437101001418?via%3Dihub> (cit. a p. 3).
- [7] Felipe Leno Da Silva e Anna Helena Reali Costa. «A survey on transfer learning for multiagent reinforcement learning systems». In: *Journal of Artificial Intelligence Research* 64 (2019), pp. 645–703. ISSN: 10769757. DOI: 10.1613/jair.1.11396 (cit. a p. 4).
- [8] Takahiro Ezaki, Daichi Yanagisawa e Katsuhiro Nishinari. «Pedestrian flow through multiple bottlenecks». In: *Physical Review E* 86.2 (30 ago. 2012). Publisher: American Physical Society, p. 026118. DOI: 10.1103/PhysRevE.86.026118. URL: <https://link.aps.org/doi/10.1103/PhysRevE.86.026118> (visitato il 28/05/2024) (cit. a p. 56).
- [9] Stephen J. Guy, Ming C. Lin e Dinesh Manocha. «Modeling collision avoidance behavior for virtual humans». In: *9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada, May 10-14, 2010, Volume 1-3*. A cura di Wiebe van der Hoek et al. IFAAMAS, 2010, pp. 575–582. URL: <https://dl.acm.org/citation.cfm?id=1838182> (cit. a p. 3).
- [10] Robert Junges e Franziska Klügl. «Programming Agent Behavior by Learning in Simulation Models». In: *Applied Artificial Intelligence* 26.4 (2012), pp. 349–375 (cit. a p. 3).

- [11] Raphael Korbmacher, Huu-Tu Dang e Antoine Tordeux. «Predicting pedestrian trajectories at different densities: A multi-criteria empirical analysis». In: *Physica A: Statistical Mechanics and its Applications* 634 (2024), p. 129440. ISSN: 0378-4371. DOI: <https://doi.org/10.1016/j.physa.2023.129440>. URL: <https://www.sciencedirect.com/science/article/pii/S0378437123009950> (cit. a p. 3).
- [12] Francisco Martinez-Gil, Miguel Lozano e Fernando Fernández. «Emergent behaviors and scalability for multi-agent reinforcement learning-based pedestrian models». In: *Simulation Modelling Practice and Theory* 74 (mag. 2017). Publisher: Elsevier B.V., pp. 117–133. ISSN: 1569190X. DOI: 10.1016/j.simpat.2017.03.003. URL: <https://linkinghub.elsevier.com/retrieve/pii/S1569190X17300503> (cit. a p. 4).
- [13] Miguel Morales. *Grokkking Deep Reinforcement Learning*. ISSN: 1098-6596 _eprint: arXiv:1011.1669v3. Manning Publications, 2020. ISBN: 978-85-7811-079-6 (cit. alle pp. 10, 11).
- [14] Sébastien Paris e Stéphane Donikian. «Activity-Driven Populace: A Cognitive Approach to Crowd Simulation». In: *IEEE Computer Graphics and Applications* 29.4 (2009), pp. 34–43 (cit. a p. 8).
- [15] Enrico Ronchi et al. *The Process of Verification and Validation of Building Fire Evacuation Models*. Gaithersburg, MD: National Institute of Standards e Technology, nov. 2013, pp. 1–85. DOI: 10.6028/NIST.TN.1822. URL: <https://nvlpubs.nist.gov/nistpubs/TechnicalNotes/NIST.TN.1822.pdf> (cit. a p. 14).
- [16] Stuart Russell e Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3^a ed. Prentice Hall, 2010 (cit. alle pp. 4–7).
- [17] Andreas Schadschneider et al. «Evacuation Dynamics: Empirical Results, Modeling and Applications». In: *Encyclopedia of Complexity and Systems Science*. A cura di Robert A. Meyers. New York, NY: Springer, 2009, pp. 3142–3176. ISBN: 978-0-387-30440-3. DOI: 10.1007/978-0-387-30440-3_187. URL: https://doi.org/10.1007/978-0-387-30440-3_187 (visitato il 16/04/2024) (cit. a p. 3).
- [18] Tobias Schrödter e The PedPy Development Team. *PedPy - Pedestrian Trajectory Analyzer*. Ver. 1.0.2. URL: <https://github.com/PedestrianDynamics/pedpy> (cit. a p. 15).
- [19] John Schulman et al. *Proximal policy optimization algorithms*. ISSN: 23318422 Publication Title: arXiv _eprint: 1707.06347. Lug. 2017. URL: <https://arxiv.org/abs/1707.06347v2> (cit. a p. 13).
- [20] Felipe Leno Da Silva e Anna Helena Reali Costa. «A survey on transfer learning for multiagent reinforcement learning systems». In: *Journal of Artificial Intelligence Research* 64 (2019), pp. 645–703. ISSN: 10769757. DOI: 10.1613/jair.1.11396 (cit. a p. 12).
- [21] Matthew E. Taylor e Peter Stone. «Transfer Learning for Reinforcement Learning Domains: A Survey». In: *Journal of Machine Learning Research* 10 (2009), pp. 1633–1685. ISSN: 18674542. DOI: 10.1007/978-3-642-27645-3_2 (cit. a p. 12).

- [22] Lisa Torrey e Jude Shavlik. «Transfer Learning». In: *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI Global, 2010, pp. 242–264. ISBN: 978-1-60566-766-9. DOI: 10.4018/978-1-60566-766-9.ch011. URL: <https://www.igi-global.com/chapter/transfer-learning/www.igi-global.com/chapter/transfer-learning/36988> (visitato il 26/06/2024) (cit. a p. 12).
- [23] Giuseppe Vizzari, Daniela Briola e Federico Pisapia. «Curriculum-Based RL for Pedestrian Simulation: Sensitivity Analysis and Hyperparameter Exploration». In: *Prestigious Applications of Intelligent Systems (PAIS-2024)*. 2024 (submitted) (cit. a p. 16).

Ringraziamenti

Desidero esprimere un sincero ringraziamento al mio relatore, il Professore Giuseppe Vizzari. La sua guida esperta, la sua disponibilità e i suoi preziosi consigli sono stati fondamentali per la realizzazione di questo lavoro. Senza il Suo supporto, questo traguardo non sarebbe stato possibile.

Un ringraziamento speciale va a Ruben. La tua collaborazione durante lo stage è stata indispensabile e sono felice che insieme abbiamo raggiunto questo importante obiettivo.

Un ringraziamento di cuore va alla mia famiglia, che mi ha sempre supportato in ogni fase del mio percorso accademico. La vostra fiducia e il vostro sostegno sono stati per me una fonte inestimabile di forza e motivazione.

Non posso non ringraziare tutti i miei amici per la compagnia, i momenti di svago e i momenti di studio trascorsi insieme. La vostra presenza ha reso più leggero e piacevole il percorso accademico.

Infine, desidero esprimere un ringraziamento speciale a Sara. Grazie per essermi stata vicina durante questo viaggio, per avermi sostenuto nei momenti difficili e per aver condiviso con me le gioie di questo percorso.

A tutti voi, va la mia più sincera gratitudine.