Alma Mater Studiorum · Università di Bologna
Campus di Cesena

SCUOLA DI INGEGNERIA E ARCHITETTURA
Corso di Laurea Magistrale in Ingegneria Informatica

SIMULAZIONI REALISTICHE DI ALGORITMI DI CROWD STEERING

Tesi di Laurea in Linguaggi e Modelli Computazionali LM

Relatore:
Chiar.mo Prof.
Mirko Viroli

Presentata da:
Luca Nenni

Correlatore:
Danilio Pianini

Terza Sessione di Laurea
Anno Accademico 2012/2013

a Gaia,
lei sa perché

Possiamo vedere solo poco davanti a noi, ma possiamo vedere tante cose che bisogna fare.

Alan Turing

Sommario

La simulazione realistica del movimento di pedoni riveste una notevole importanza nei mondi dell'architettonica e della sicurezza (si pensi ad esempio all'evacuazione di ambienti), nell'industria dell'entertainment e in molti altri ambiti, importanza che è aumentata negli ultimi anni.
Obiettivo di questo lavoro è l'analisi di un modello di pedone esistente e l'applicazione ad esso di algoritmi di guida, l'implementazione di un modello più realistico e la realizzazione di simulazioni con particolare attenzione alla scalabilità.
Per la simulazione è stato utilizzato il framework Alchemist, sviluppato all'interno del laboratorio di ricerca APICe, realizzando inoltre alcune estensioni che potranno essere inglobate nel pacchetto di distribuzione del sistema stesso.
I test effettuati sugli algoritmi presi in esame evidenziano un buon guadagno in termini di tempo in ambienti affollati e il nuovo modello di pedone risulta avere un maggiore realismo rispetto a quello già esistente, oltre a superarne alcuni limiti evidenziati durante i test e ad essere facilmente estensibile.

Indice

Introduzione

Questo lavoro si pone come obiettivo principale l'analisi di algoritmi di guida di pedoni, in particolare di un algoritmo, detto *crowd sensitiveness*, per mezzo di simulazioni realistiche di movimento, in particolare in ambienti indoor. Durante la realizzazione si è inoltre resa necessaria l'implementazione nell'ambiente utilizzato di un nuovo modello di pedone e di alcuni elementi di supporto che sono stati integrati nel framework. Fondamentale per questa analisi è, come si è detto, avere una simulazione che sia realistica e scalabile, in grado di essere applicata ad ambienti dei più disparati e a numeri anche molto elevati di individui, senza soffrirne troppo in termini di performance. Tre sono gli elementi fondamentali per questo studio:

- un modello realistico di pedone;
- uno o più algoritmi di guida (crowd steering);
- un tool di simulazione.

Gli ambiti di applicazione sono diversi e con il passare degli anni stanno acquisendo maggiore importanza. Come primo esempio si porta il caso dell'evacuazione di ambienti, che con l'aiuto della simulazione può studiare le situazioni di pericolo e i probabili comportamenti degli individui, per poi proporre strategie efficaci ed efficienti per guidare il deflusso delle persone. Attraverso gli stessi strumenti è poi possibile verificare la validità di queste strategie, testarle in situazioni differenti e analizzarne punti di forza e criticità. Studi sui disastri con coinvolgimento di folle avvenuti negli ultimi 150 anni (cfr. [2]), avvenuti per la maggior parte in luoghi di divertimento, stadi, concerti, o luoghi di culto, hanno evidenziato i comportamenti tipici delle persone in condizioni normali e di panico e proposto modelli diversi, ad esempio basati su code o su matrici di transizione, o modelli stocastici. Più recentemente questi modelli si sono evoluti considerando ad esempio meccanismi di auto-organizzazione di gruppi di persone e tentando di schematizzare questi comportamenti in alcune regole, come ad esempio la tendenza dei pedoni ad organizzarsi in corsie in ambienti piuttosto affollati, oppure l'accalcarsi intorno alle uscite più note trascurando quelle più lontane o nascoste, ma allo stesso tempo più sicure, in condizioni di panico. Un tipo di modellazione piuttosto diffuso, che è stato utilizzato nei due modelli presentati in questo lavoro, concepisce l'influenza di ciò che sta attorno al pedone (ostacoli, pareti, altri individui) sul pedone stesso come una sommatoria di forze, alcune attrattive, come ad esempio quella che spinge la persona verso il proprio obiettivo, altre repulsive, come quelle esercitate dagli ostacoli. Un sistema adatto a rappresentare l'inseguimento di un obiettivo, che può essere influenzato da algoritmi di guida, è quello della diffusione di un gradiente, che genera nell'ambiente un campo di vettori che portano verso i punti di interesse. Se, ad esempio, il valore riportato nel campo dipende dalla distanza dall'obiettivo, il pedone può scegliere il percorso più conveniente; allo stesso modo, intervenendo su queste grandezze, è possibile influenzare il movimento portando

il pedone a modificare la propria rotta, è questo il principio alla base del crowd steering. Riguardo ai tre elementi succitati, come ambiente di simulazione è stato scelto Alchemist, un framework per la modellazione di sistemi complessi che fra le proprie incarnazioni ne offre una (SAPERE) adatta allo studio del movimento di pedoni. Questa inoltre contiene un modello già definito di pedone, anche se come vedremo si deciderà di implementarne uno nuovo, e un algoritmo di crowd steering basato sul numero di pedoni che si trova all'interno di un'area.

L'organizzazione di questo lavoro è la seguente:

- In questa parte introduttiva si è mostrato lo scenario e l'ambito del lavoro, con la descrizione del problema trattato.
- Nel capitolo 1 si introdurrà il framework utilizzato per l'implementazione del modello e la realizzazione delle simulazioni, riassumendone in breve la struttura, le caratteristiche peculiari e gli elementi fondamentali.
- Nel capitolo 2 verrà descritto lo stato iniziale del lavoro, facendo riferimento al modello di pedone già implementato, ai test effettuati e alle simulazioni realizzate; verranno quindi analizzati i limiti emersi durante questa fase, esplicitando le motivazioni che hanno portato alla definizione di un nuovo modello di pedone.
- Nel capitolo 3, cuore di questo lavoro, sarà descritto il nuovo modello di pedone implementato, le modifiche apportate al modello in seguito alle prime prove effettuate, i limiti trovati e le possibili soluzioni.
- Nel capitolo 4 si mostreranno i test effettuati sul nuovo modello e i punti di forza trovati.
- Nel capitolo 5 si analizzeranno gli algoritmi di guida, in particolare la crowd sensitiveness.
- Nel capitolo 6 verranno tratte le conclusioni sul lavoro svolto e proposte alcune prospettive per ulteriori sviluppi.

1 Alchemist

In questo capitolo sarà introdotto il framework utilizzato per la simulazione, la sua struttura e gli elementi principali che lo compongono. Verrà poi descritta l'incarnazione SAPERE, cui si è fatto riferimento in questo lavoro, il linguaggio specifico dell'incarnazione (con un esempio di codice), con alcuni brevi cenni sulle classi disponibili nella distribuzione di Alchemist.
Alchemist (rif. [8]) è un framework per la modellazione e simulazione di sistemi complessi il cui obiettivo è colmare il vuoto fra gli algoritmi stocastici di simulazione e i simulatori basati su sistemi multi agente, costituendo un ibrido fra le due tipologie. Il progetto, a cura del laboratorio di ricerca APICe, è realizzato in Java e rilasciato sotto licenza GPL.

1.1 Incarnazioni

A seconda della struttura utilizzata per definire la concentrazione è possibile avere diverse incarnazioni del simulatore, ciascuna per un utilizzo specifico. Tutte condividono però alcuni elementi fondamentali, quali il motore di simulazione e il linguaggio Alchemist. Per ogni incarnazione è possibile definire azioni, reazioni e nodi specifici, e utilizzando xText si può creare un linguaggio specifico. Questo è ciò che è avvenuto ad esempio per l'incarnazione SAPERE, il cui obiettivo principale è la simulazione di ecosistemi pervasivi. La struttura fondamentale è la tupla di dati, e può raggiungere un livello di complessità piuttosto elevato. In questa tesi verrà utilizzata questa incarnazione del simulatore, opportunamente estesa, per la realizzazione di simulazioni realistiche del movimento di pedoni e lo studio di algoritmi di guida.

1.1.1 L'incarnazione SAPERE

Elementi fondamentali della simulazione

Gli elementi che costituiscono il modello computazionale del simulatore sono i seguenti (da [8]):

Ambiente
lo spazio che contiene i nodi e all'interno del quale si svolge la simulazione.

Linking rule
la regola che definisce in che modo i nodi stabiliscono connessioni o le interrompono (vincoli numerici, spaziali, presenza di ostacoli, ecc.).

Nodi
elementi programmabili che possono contenere molecole e muoversi all'interno dell'ambiente, e all'interno dei quali possono avvenire reazioni.

Molecole
ciascuna con un proprio valore di concentrazione.

Reazioni
definite da una serie di condizioni riguardanti l'ambiente, una velocità che può cambiare al variare di queste condizioni e un insieme di azioni che rappresentano l'effetto della reazione stessa.

Per descrivere la sintassi e le parole chiave del linguaggio DSL utilizzato nell'incarnazione SAPERE sarà utilizzato il codice di una simulazione di esempio, la stessa utilizzata per i test del capitolo 4. Il file del codice deve avere estensione .alsap. Come primo punto occorre descrivere l'environment:

Listing 1.1: Environment

```
1 environment sPietro
2 type PngEnvironment
3 params "15,C:/eclipse/runtime-EclipseApplication/alchemist-
    simulations/src/plants/spietro.png"
4 with linking rule SelectiveEuclideanDistanceWithObstacles params
    "20,0,infrastructure"
5 with random seed 950632084
```

Analizzando passo passo questa prima parte otteniamo le seguenti informazioni:

1. `environment sPietro`: la parola chiave indica l'inizio della specifica e definisce il nome dell'ambiente (opzionale).

2. `type PngEnvironment`: il tipo dell'ambiente PngEnvironment indica che esso è caricato da un file immagine in formato PNG. L'immagine viene partizionata in rettangoli in base al colore, considerando solo il nero o gli altri indicati (opzionali).

3. `params 15,C:/eclipse/runtime-EclipseApplication/alchemist-simulations/src/plants/spietro.png`: i parametri passati al costruttore sono il livello di zoom e il percorso completo del file immagine.

4. `with linking rule SelectiveEuclideanDistanceWithObstacles params 20, 0,infrastructure`: come detto precedentemente, la linking rule è la regola che definisce quali collegamenti debbano essere stabiliti fra i nodi. In questo specifico caso si tratta di una regola che collega solo a un numero massimo specificato di nodi, che contengono una determinata molecola, all'interno di un determinato raggio e fra i quali non si frappongano ostacoli. I parametri passati sono il range di ricerca (20 metri), il numero massimo di link per nodo (settato a 0, quindi illimitato) e la molecola da ricercare. Gli unici link possibili dunque sono fra e con nodi che contengono molecole "infrastructure".

5. `with random seed 950632084`: il seed del motore random non è un parametro necessario, ma garantisce la riproducibilità di una simulazione.

Vengono poi definiti gli LSA sotto forma di tuple, in stile Linda:

Listing 1.2: LSA

```
1 lsa source <source, Type, Distance>
2 lsa target <source, target, 0>
3 lsa gradient <grad, Type, Distance>
4 lsa crowd <crowd, L>
5 lsa red <red>
6 lsa green <green>
7 lsa blue <blue>
8 lsa infrastructure <infrastructure>
9 lsa person <person>
```

Gli ultimi 5, costituiti da una sola stringa, servono a definire i tipi di nodo (persona o infrastruttura) o la sua colorazione nella GUI Alchemist. Un interesse maggiore, invece, hanno le prime righe: esse sono utilizzate per la propagazione del gradiente, che permette ai pedoni di raggiungere un obiettivo, e per il sensing della folla. Concretamente possono tradursi in sensori di posizione o movimento, access point wireless, monitor che indicano una direzione. Segue poi il posizionamento dei nodi, in questo caso sono stati riportati prima i nodi di infrastruttura:

Listing 1.3: Nodi infrastruttura

```
1 place node at point (144,20)
2 containing red target infrastructure
3 with reactions
4 reaction SAPEREGradient params "ENV,NODE,RANDOM,source,gradient,2,((
     Distance+#D)+(0.5*L)),crowd,2000000,10" []-->[]
5 eco-law compute_crowd []-1-> [agent CrowdSensor params "ENV,NODE"]
6
```

```
7  place node at point (144,133)
8  containing red target infrastructure
9  with reactions
10 reaction SAPEREGradient params "ENV,NODE,RANDOM,source,gradient,2,((
       Distance+#D)+(0.5*L)),crowd,2000000,10" []-->[]
11 eco-law compute_crowd []-1-> [agent CrowdSensor params "ENV,NODE"]
12
13 place node at point (37,102)
14 containing red target infrastructure
15 with reactions
16 reaction SAPEREGradient params "ENV,NODE,RANDOM,source,gradient,2,((
       Distance+#D)+(0.5*L)),crowd,2000000,10" []-->[]
17 eco-law compute_crowd []-1-> [agent CrowdSensor params "ENV,NODE"]
18
19 place node at point (37,77)
20 containing red target infrastructure
21 with reactions
22 reaction SAPEREGradient params "ENV,NODE,RANDOM,source,gradient,2,((
       Distance+#D)+(0.5*L)),crowd,2000000,10" []-->[]
23 eco-law compute_crowd []-1-> [agent CrowdSensor params "ENV,NODE"]
24
25 place node at point (37,53)
26 containing red target infrastructure
27 with reactions
28 reaction SAPEREGradient params "ENV,NODE,RANDOM,source,gradient,2,((
       Distance+#D)+(0.5*L)),crowd,2000000,10" []-->[]
29 eco-law compute_crowd []-1-> [agent CrowdSensor params "ENV,NODE"]
30
31 place 117 nodes in rect (54,40,120,92) interval 10
32 containing in all green infrastructure
33 with reactions
34 reaction SAPEREGradient params "ENV,NODE,RANDOM,source,gradient,2,((
       Distance+#D)+(0.5*L)),crowd,2000000,10" []-->[]
35 eco-law compute_crowd []-1-> [agent CrowdSensor params "ENV,NODE"]
```

Come si può notare, i primi 5 contengono il target e saranno colorati in rosso, mentre gli altri, rappresentati in verde, saranno i nodi di infrastruttura veri e propri. Le parti di interesse si possono riassumere nelle modalità di posizionamento dei nodi:

1. `place node at point (144,20)`: il singolo nodo ha le coordinate indicate.
2. `place 117 nodes in rect (54,40,120,92) interval 10`: un set di nodi è di-

sposto all'interno di un'area di una forma specifica, in questo caso rettangolare, e a distanza di 10 metri l'uno dall'altro. Opzionalmente è possibile impostare una tolleranza entro la quale il posizionamento del nodo deve avvenire.

e nelle reazioni che avvengono al loro interno:

1. `reaction SAPEREGradient params ENV,NODE,RANDOM,source,gradient,2, ((Distance+#D)+(0.5*L)),crowd,2000000,10 []-->[]`: La reazione gradiente ha come parametri l'ambiente stesso, il nodo, il motore random, la molecola di origine e quella di gradiente, la posizione all'interno della tupla in cui inserire il valore, l'espressione per il calcolo, una tupla di template che può essere utilizzata per modificare il calcolo (cfr. punto seguente), una soglia massima oltre la quale il gradiente evapora e il rate Markoviano.

2. `eco-law compute_crowd []-1-> [agent CrowdSensor params ENV,NODE]`: l'agente *CrowdSensor* permette di modificare il valore del gradiente in base al numero di pedoni che si trovano collegati al nodo in questione. Questo permette di evitare aree molto affollate in favore di percorsi più lunghi ma liberi.

L'immagine 1.1 rappresenta l'ambiente di simulazione, i nodi di infrastruttura e i link stabiliti fra di essi, come definiti dal codice fin qui riportato. Nella figura 4.1 a pagina 34 sono stati inseriti anche i pedoni. Si trovano infine nel codice i nodi rappresentanti i pedoni:

Listing 1.4: Pedoni

```
1 place 400 nodes in rect (54,48,120,60)
2 containing in all person blue
3 with reactions
4 []-1->[agent GradientFollowingPredictivePedestrian params "ENV,NODE,
     RANDOM,REACTION,gradient,2,3,20"]
```

La reazione

1. `[]-1->[agent GradientFollowingPredictivePedestrian params ENV,NODE, RANDOM,REACTION,gradient,2,3,20]`: contiene l'agente *GradientFollowingPredictivePedestrian*, cioè il pedone predittivo (oggetto di questa tesi), che ha come parametri, oltre ai primi citati in precedenza, la reazione stessa, la molecola da cercare, la posizione del valore all'interno della tupla, il livello di attenzione e il range di ricerca visiva. Maggiori dettagli sul senso di questi valori si troveranno nel capitolo apposito.

Saranno descritte nelle prossime pagine le principali implementazioni di questi elementi, già presenti nella distribuzione di Alchemist o implementati per rispondere alle necessità delle nuove simulazioni.

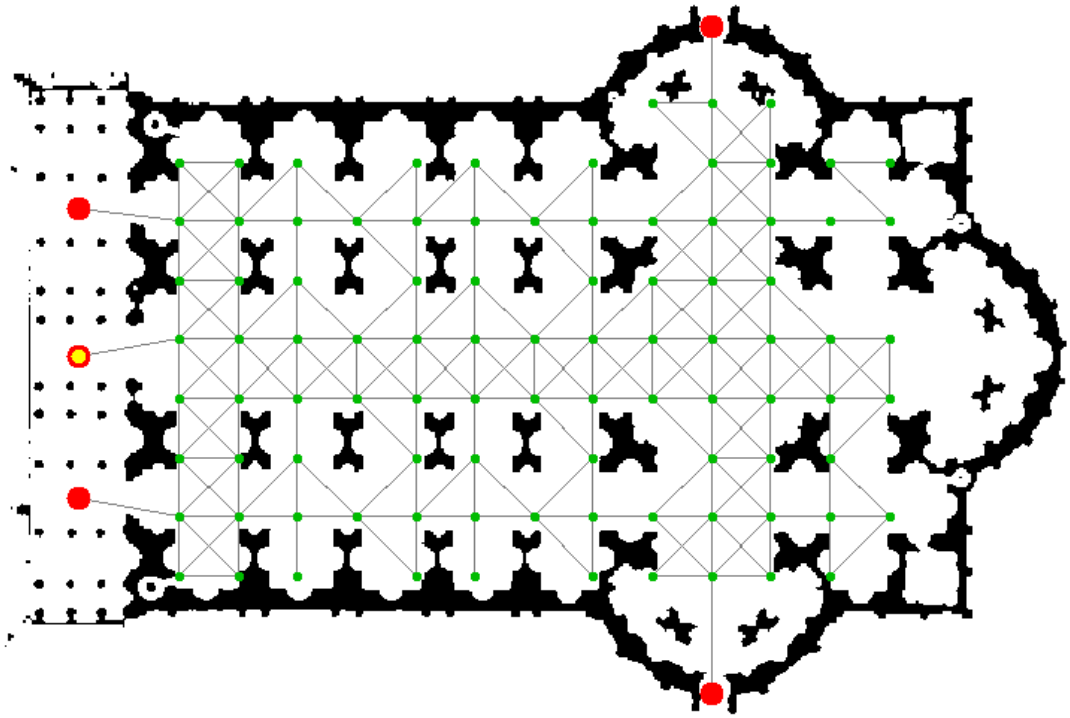

Figura 1.1: *L'ambiente di simulazione e i nodi di infrastruttura*

1.2 Implementazioni

Le implementazioni dei vari elementi sono costituite da classi Java, nelle quali vengono mappati gli oggetti definiti all'interno del file .alsap di descrizione della simulazione. Ciascun tipo di oggetto appartiene al package `it.unibo.alchemist.model.implementations.`*tipo* (es. *it.unibo.alchemist.model.implementations.linkingrules*)

1.2.1 Environments

Gli ambienti sono basati sulle classi astratte riportate di seguito:

AbstractEnvironment una prima implementazione di base che permette di gestire un set di nodi e la loro posizione.

AbstractLinkingRuleEnvironment che estende la precedente inserendo una linking rule, quindi la possibilità che i nodi contenuti si colleghino fra loro.

Seguono poi le prime implementazioni complete di ambienti utilizzabili all'interno di simulazioni:

Continuous2DEnvironment implementa i metodi astratti indicati nelle classi precedenti per popolare l'environment, ottenerne le dimensioni e muovere i nodi.

LimitedContinuos2D estende la precedente dando all'ambiente limiti spaziali, che impediscono ai nodi di essere mossi in posizioni che non sono permesse.

Continuous2DObstacles aggiunge la possibilità di inserire ostacoli all'interno dell'environment.

A partire da quest'ultima classe è possibile definire nuovi ambienti come nuove classi Java, inserendo a mano i rettangoli che andranno a costituire gli ostacoli. Questa tecnica in realtà è al momento deprecata, dato che è stata sostituita dalla costruzione dell'environment a partire da un file immagine in formato PNG, che viene scomposto in più rettangoli a seconda dei colori, che vanno a costituire gli ostacoli. Di questo si occupa la classe *PngEnvironment*, che estende *Continuous2DObstacles* e accetta i seguenti parametri:

- il path completo dell'immagine (unico parametro non opzionale)
- il fattore di zoom in percentuale
- la lista di triple di interi (R,G,B) rappresentanti i colori che devono essere considerati ostacoli
- il valore delta che indica il movimento in diagonale della mappa nell'ambiente
- un booleano che indica se la copertura deve essere minima o trovata tramite euristiche

1.2.2 Linking rules

Come già detto, le linking rules sono le regole che determinano la creazione di collegamenti fra i nodi presenti nell'ambiente. Questi permetteranno, ad esempio, la diffusione di informazioni o di un gradiente. La linking rule permette inoltre a un nodo di conoscere il proprio vicinato (*neighborhood*), cioè l'insieme dei nodi ai quali è collegato. Nella distribuzione di Alchemist sono già presenti le seguenti linking rules:

NoLinks è la regola più semplice: nessun collegamento viene creato e il neighborhood di ogni nodo è costituito da una lista vuota di elementi.

EuclideanDistance ha come parametro un numero che indica un range massimo: il neighborhood è costituito da tutti i nodi che si trovano all'interno della circonferenza di raggio indicato, quindi a una distanza euclidea massima "range", dal nodo in esame.

AdaptiveRange è costituito dai seguenti parametri:

- raggio di default
- numero di nodi
- tolleranza
- range minimo
- range massimo
- aggiustamento

di cui gli ultimi tre sono opzionali. La regola è un'estensione della precedente, in cui il range può variare di uno step pari al parametro di aggiustamento, fra un minimo e un massimo, a partire da un valore di default, finché il numero di nodi all'interno del raggio non raggiunge il numero prefissato; se questo viene superato di un numero di elementi pari alla tolleranza il range cala, mentre aumenta se i nodi del vicinato sono meno del numero prefissato meno il valore della tolleranza.

ObstaclesBreakConnection è basata anch'essa sulla distanza euclidea, ma in un ambiente che contiene ostacoli questi ultimi interrompono i link che li attraversano.

A partire da queste, sono state create le seguenti nuove regole:

SelectiveAdaptiveRange come la regola su cui è basata collega i nodi all'interno di un range variabile, ma ha come ulteriore parametro un template di molecola: i link vengono creati solo se almeno uno dei due nodi collegati contiene la molecola indicata.

SelectiveEuclideanDistance anche questa è basata sulla distanza euclidea, nuovamente in maniera selettiva.

SelectiveEuclideanDistanceWithObstacles è la regola più complessa, derivata dalla precedente ma in un ambiente con ostacoli.

1.2.3 Reactions

Le principali reazioni che è possibili inserire nel sistema sono:

LsaTrigger reazione che cerca di avviarsi in un momento specificato e viene poi rimossa dalla simulazione

LsaStaticFixedTimeReaction che avviene ciclicamente alla scadenza di un intervallo di tempo

SAPEREGradient che implementa un sistema di gradiente stabile e veloce, che è stato utilizzato per le simulazioni trattate in questo lavoro.

1.2.4 Actions

Le azioni rappresentano gli agenti che si occupano dell'inserimento, del movimento e della rimozione di nodi. In questa specifica incarnazione, in particolare, e in questo lavoro, saranno trattate principalmente le actions che rappresentano modelli di pedone e algoritmi per lo steering. Non sono trattati in questo capitolo ma saranno affrontati approfonditamente nei prossimi.

2
Modello iniziale: SocialForce

In questo capitolo verranno mostrati gli elementi già disponibili all'inizio del lavoro, le simulazioni realizzate, i test effettuati e i limiti trovati, dando particolare rilievo alla logica di calcolo del movimento del pedone, che sarà successivamente messa a confronto con quella del nuovo modello. Saranno inoltre fatti alcuni accenni al tool utilizzato per il model-checking.

2.1 Il modello utilizzato

Il modello di pedone utilizzato, presente attualmente nella distribuzione di Alchemist, è realizzato da C. Casalboni e M. Bombardi ed è basato sul lavoro di D. Helbing, L. Buzna, A. Johansson e T. Werner ([2]). Parte di questo capitolo è adattata dal manuale di Alchemist ([8]), in cui il modello stesso è descritto in maniera approfondita. Si tratta di un modello basato su vettori, in cui le interazioni fra pedoni e la scelta della direzione da seguire è modellata per mezzo di due tipi di forze:

- una repulsiva, esercitata dalle pareti, dagli ostacoli e dagli altri pedoni che gli agenti incontrano lungo il loro percorso
- una attrattiva, generata dall'obiettivo

2.1.1 Calcolo della velocità

Nel calcolo della velocità del pedone entrano in gioco quattro differenti forze: una prima forza generata dall'inseguimento del target, una seconda forza repulsiva fra pedoni che evita le collisioni, una terza che li spinge ad evitarsi, e un'ultima forza generata dagli ostacoli. A ogni step della simulazione si calcola l'accelerazione come somma pesata di

queste quattro forze e da questa e dal valore precedente viene computata la nuova velocità. Si pone ovviamente un limite massimo alla velocità massima che il pedone può avere.

2.1.2 Inseguimento dell'obiettivo

Il target esercita l'unica forza di tipo attrattivo di quelle che influiscono sul pedone. Viene scelto come obiettivo il nodo fra quelli nel vicinato del pedone che ha il valore del gradiente inferiore. Questo è contenuto all'interno di una molecola che generalmente è distribuita in un insieme di nodi fissi, detti di infrastruttura, ma in alcune simulazioni si trova anche all'interno dei nodi rappresentanti i pedoni stessi. Le componenti di questa forza sono proporzionali alla distanza fra il pedone e il nodo target, secondo queste formule:

$$desiredForce_x = \frac{distance_x}{distance}$$
$$desiredForce_y = \frac{distance_y}{distance}$$

in cui $distance_x$ e $distance_y$ rappresentano le componenti orizzontale e verticale della distanza euclidea fra i due nodi, mentre $distance$ è la distanza stessa.

2.1.3 Avoidance degli ostacoli

La prossima forza in esame è generata dagli ostacoli incontrati lungo il cammino. Dati tutti gli ostacoli che il pedone può vedere, cioè quelli compresi all'interno di un dato range, si prende in esame il più vicino al nodo interessato e si calcola la forza repulsiva dal centro di questo ostacolo, per questo motivo il pedone è soggetto a una forza che è tanto più grande quanto è maggiore la sua distanza dal centro dell'ostacolo; la scelta dipende dalla necessità di avere un contributo corretto anche nel caso di pareti non quadrate (come già ricordato, ogni ostacolo è rappresentato da una somma di rettangoli). La formula per il calcolo di quest'ultima forza è simile a quelle già note:

$$obstacleForce_x = -obstacleForceStrength * \frac{distance_x}{distance}$$
$$obstacleForce_y = -obstacleForceStrength * \frac{distance_y}{distance}$$

dove ovviamente $obstacleForceStrength$ rappresenta l'intensità della forza, $distance_x$ e $distance_y$ le componenti sui due assi della distanza fra il pedone e il centro dell'ostacolo e $distance$ l'effettiva distanza euclidea fra il nodo e il centro.

2.1.4 Avoidance dei pedoni

Forza sociale

La forza sociale è la forza repulsiva fra pedoni che fa sì che si evitino collisioni. Vi è per ognuno uno spazio vitale che in condizioni normali non deve essere violato. Se un pedone j invade lo spazio vitale di un altro pedone i, trovandosi quindi a una distanza inferiore a un determinato raggio di interazione, la forza repulsiva si calcola secondo la seguente formula:

$$\vec{F_{ij}^{rep}} = -m_i k_{ij} \frac{(\eta v_i^0 + v_{ij})^2}{d_{ij}} \vec{e_{ij}}$$

Nella formula:

- m_i rappresenta la massa del pedone i-esimo;
- k_{ij} ha per effetto la riduzione del raggio di effetto della forza repulsiva al solo angolo visivo del pedone e si calcola secondo questa formula:

$$k_{ij} = \begin{cases} \frac{(\vec{v_i} \cdot \vec{e_{ij}})}{\|\vec{v_i}\|}, & \text{se } \vec{v_i} \cdot \vec{e_{ij}} > 0 \text{ e } \|\vec{v_i}\| \neq 0 \\ 0, & \text{diversamente} \end{cases}$$

- η è il valore della forza sociale; nell'implementazione è stato scelto un valore di 0.2.
- d_{ij} è l'effettiva distanza fra i pedoni i e j.
- v_{ij} è la velocità relativa, definita in modo che i pedoni più lenti siano meno influenzati dalla presenza di altri più veloci davanti a sé:

$$v_{ij} = \begin{cases} (\vec{v_i} - \vec{v_j}) \cdot \vec{e_{ij}}, & \text{se } (\vec{v_i} - \vec{v_j}) \cdot \vec{e_{ij}} > 0 \\ 0, & \text{diversamente} \end{cases}$$

- e_{ij} è la direzione della forza fra i due pedoni.

Forza di schivata

Questa rappresenta la forza che consente a un pedone di girare a destra o a sinistra se si trova a una distanza inferiore a un dato range da un'altra persona. Questo valore è settato a 0.5 metri. A seconda del Paese in cui si è cresciuti, si può preferire superare un ostacolo aggirandolo da destra o da sinistra, come avviene ad esempio nel caso dei Paesi britannici (così come per le loro regole di guida), mentre nel resto dell'Europa si tende a superare gli ostacoli da destra. La soluzione trovata, in particolare, è di variare la direzione di ± 45 gradi secondo questa formula:

- Virata a sinistra:

$$dodgeForce_x = -dodgeForceStrength * desiredForce_y$$
$$dodgeForce_y = dodgeForceStrength * desiredForce_x$$

- Virata a destra:

$$dodgeForce_x = dodgeForceStrength * desiredForce_y$$
$$dodgeForce_y = -dodgeForceStrength * desiredForce_x$$

Dove $dodgeForceStrength$ è un fattore che rappresenta l'intensità della forza, cioè la probabilità (in percentuale) che il pedone scelga di girare a destra.

2.2 Implementazione in Alchemist

Il pedone è stato implementato all'interno del progetto riguardante l'incarnazione SAPERE, estendendo alcune classi presenti all'interno del package *it.unibo.alchemist.model .implementations.actions*. In particolare le logiche di calcolo delle forze precedentemente indicate sono contenute all'interno di una classe *SocialForceAgent*, che estende le seguenti:

- una classe astratta *SAPEREMoveNodeAgent* che implementa le interfacce che permettono a un nodo di muoversi all'interno dell'ambiente, che a sua volta estende:
- una classe astratta *SAPERELocalAgent*, che identifica un agente che non effettua azioni sul proprio neghborhood; questa estende
- una classe *SAPEREAgent*, anch'essa generica, creata sulla base di altre classi di azioni astratte.

Come detto, *SocialForceAgent* implementa il calcolo delle varie forze attrattive e repulsive, ma si tratta a sua volta di una classe astratta, che viene concretizzata in due altre classi, le quali differiscono per il valore di alcune costanti: si tratta di *SocialForceAsianAgent* e *SocialForceEuropeanAgent*, che implementano rispettivamente un pedone dal comportamento tipico di un asiatico o anglosassone e un pedone dal comportamento europeo (che supera quindi un ostacolo preferibilmente passandovi a destra). Le stesse convenzioni di coding saranno utilizzate, come si vedrà nel prossimo capitolo, anche nella ri-modellazione del pedone.

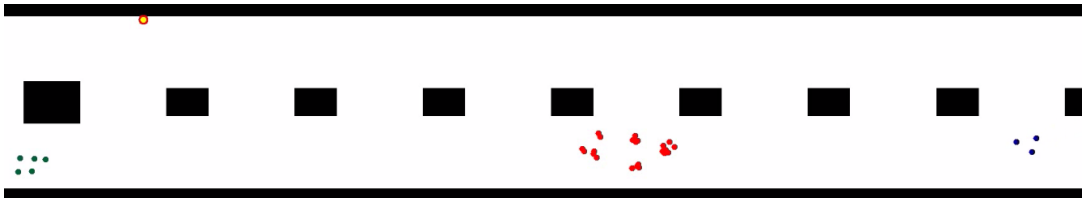

Figura 2.1: *La simulazione "Fiera" alla partenza*

2.3 Simulazioni generate e test effettuati

Questo modello è stato utilizzato nella realizzazione di alcune simulazioni per una presentazione di Alchemist e per testare la bontà del sistema di steering. Ci si è concentrati in particolare nella simulazione di un ambiente di una fiera, comprendente un'unica sala molto lunga con alcuni stand rettangolari nella parte centrale, e diversi gruppi di pedoni fermi e in movimento. Obiettivo principale dei test è stato ottenere una simulazione realistica del movimento di un gruppo di pedoni in un ambiente affollato, da un estremità all'altra dell'ambiente, verificando inoltre il guadagno ottenuto grazie agli algoritmi di steering. Si è trattato quindi, fondamentalmente, di partire da una simulazione base con alcuni pedoni da guidare, un piccolo gruppo di persone ferme a fare da ostacolo e un target situato alle spalle di questi, e di verificare il comportamento dei pedoni guidati in caso di steering abilitato e disabilitato. Successivamente per gradi si sono aggiunti dettagli e complicazioni, in modo che la simulazione approssimasse un caso reale o realistico il più possibile.

Già partendo da questa simulazione (figura 2.1) è possibile verificare l'efficacia del sistema di steering dei pedoni: quando questo è disabilitato, infatti, il gruppo in movimento percorre la strada più corta verso il target, e si trova a dover passare fra le persone poste come ostacolo: in questa zona si è costretti a rallentare per evitare collisioni, mentre nel caso in cui il sistema è attivato i pedoni in movimento percorrono una strada più lunga, passando nella parte superiore dell'ambiente, ma mantenendo una buona velocità. Registrando le due simulazioni con un software di cattura schermo e montando i due filmati in split-screen è possibile apprezzare meglio questo risultato; l'immagine 2.2 sintetizza efficacemente quanto scritto nell'ultimo paragrafo: i pedoni guidati (nella parte superiore dell'immagine) nello stesso istante di tempo si trovano, nonostante la simulazione sia piuttosto semplice, in vantaggio rispetto a quelli non guidati.

La stessa simulazione è stata ampliata aggiungendo via via nuovi gruppi di pedoni ostacolo, sia fissi che in movimento, andando ad analizzare volta per volta il comportamento con e senza steering.

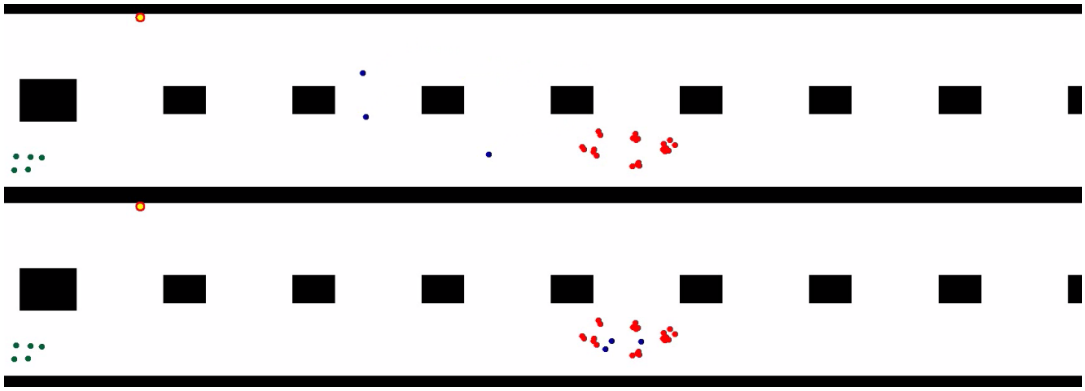

Figura 2.2: *Steering abilitato e disabilitato: simulazioni a confronto*

2.4 APMC

Particolare rilevanza per il calcolo della bontà delle simulazioni, soprattutto da un punto di vista quantitativo, ha avuto lo strumento di model checking approssimato realizzato da D. Ensini e presente anch'esso nella distribuzione di Alchemist.

2.4.1 Il concetto di model checking approssimato

A differenza del model checking vero e proprio, il quale analizza tutti gli stati possibili di un sistema per verificarne alcune proprietà (come ad esempio la safety) e restituisce un risultato esatto, un tool approssimato lancia un numero elevato di simulazioni del sistema in esame e ha come risultato un intervallo di confidenza di una certa dimensione della proprietà studiata. Il numero elevato dei possibili stati, infatti, renderebbe impossibile la verifica di tutti gli stati del sistema. Un esempio di possibile risultato potrebbe essere il numero di nodi che raggiungono l'obiettivo entro un tempo determinato, o al contrario il tempo necessario affinché una data percentuale di nodi si trovi entro un certo raggio dai target.

2.4.2 Test effettuati e risultati

I test effettuati tramite il tool di model checking riguardano principalmente i meccanismi di guida dei pedoni, e saranno perciò descritti approfonditamente e analizzati nel capitolo dedicato agli algoritmi di crowd steering.

2.5 Limiti

Il principale limite emerso durante i test è costituito dalle modalità in cui vengono percepiti o evitati gli ostacoli. La presenza di pedoni nell'intorno infatti è verificata tramite i links, per cui quelli nascosti da ostacoli sono completamente ignorati in presenza di linking rule che interrompono i collegamenti. Se ad esempio a causa della forza causata dalla presenza di altri pedoni il nodo si viene a trovare a ridosso di una parete, vi resta bloccato e perde la percezione degli altri nodi. Il problema del blocco dei pedoni sulle pareti in particolare ha portato a ritenere più adatto un sistema di percezione basato sul raggio visivo, lasciando al sistema basato sui links e sul gradiente la parte di steering, motivo per cui si è deciso di implementare un nuovo modello.

3
Il pedone predittivo

In questo capitolo sarà descritto il nuovo modello di pedone che si è deciso di utilizzare, le sue particolarità, i punti di forza e quelli deboli, la sua implementazione in Alchemist e i test cui è stato sottoposto.

3.1 Il modello utilizzato

Per l'implementazione in Alchemist del nuovo pedone si è fatto riferimento al modello descritto da Ioannis Karamouzas, Peter Heil, Pascal van Beek and Mark H. Overmars in A Predictive Collision Avoidance Model for Pedestrian Simulation ([4]). I concetti fondamentali su cui il modello si basa sono i seguenti:

Principio del minimo sforzo si assume che il pedone segua fra le possibili traiettorie quella più efficiente a livello energetico, prendendo in considerazione quindi la lunghezza del percorso e il numero di deviazioni (a meno, ovviamente, delle interazioni con gli altri pedoni).

Scansione ed esternalizzazione il pedone segnala con il proprio linguaggio corporeo la direzione che intende seguire, e allo stesso tempo cerca intorno a sé informazioni analoghe per evitare collisioni, che vengono risolte tramite una coordinazione volontaria.

Spazio personale concetto molto importante nelle interazioni sociali, assume che intorno a ogni persona vi sia uno spazio che non deve essere invaso da altri individui, cioè la distanza minima cui essi devono tenersi affinché la persona non si senta a disagio.

3.1.1 Differenze e similarità rispetto al modello precedente

La principale differenza fra il pedone predittivo e quello basato sulla *social force* risiede, a livello di modello, principalmente nella modalità con cui vengono evitate le collisioni con gli altri pedoni. Vi sono poi alcuni dettagli e miglioramenti realizzati a livello di implementazione che lo rendono utilizzabile in contesti più ampi e meno dipendente dalla distribuzione dei nodi di infrastruttura. Nel riportare in codice il modello si è prestata particolare attenzione all'estensibilità, realizzando quindi un elemento funzionante ma aperto a integrazioni e miglioramenti. Come nel modello precedente, invece, la variazione di velocità e di direzione del pedone è espressa come conseguenza di una sommatoria di forze, di cui una con effetto positivo (verso l'obiettivo) e le altre in senso negativo (avoidance degli altri pedoni e degli ostacoli). Inoltre, come si vedrà, alcuni concetti sono ripresi proprio dal modello di Helbing, di cui questo si propone come evoluzione.

3.1.2 Inseguimento dell'obiettivo

Nel modello si assume che il pedone abbia una velocità preferita u_i^{pref} che tende a raggiungere gradualmente in un tempo τ, nella direzione del proprio obiettivo g_i, a partire dalla posizione attuale x_i e dalla velocità attuale $\vec{v}$). Sul pedone agisce quindi una forza $\vec{F_g}$, la cui formula è la seguente:

$$\vec{F_g} = \left(\frac{1}{\tau}\right)(u_i^{pref}\vec{n_{gi}} - \vec{v}), \text{ dove}$$
$$\vec{n_{gi}} = \frac{g_i - x_i}{\|g_i - x_i\|}$$

La formulazione della forza come dipendente dalla distanza dal target è simile a quella descritta nel modello *social force* di Helbing.

3.1.3 Avoidance degli ostacoli

Le pareti presenti nell'ambiente, in particolare, quelle che si trovano entro in certo raggio, esercitano sul pedone una forza repulsiva che è sintetizzata nella formula:

$$\vec{F_w} = \begin{cases} \vec{n_w}\frac{d_s + r_i - d_{iw}}{(d_{iw} - r_i)^\kappa}, & \text{se } d_{iw} - r_i < d_s \\ 0, & \text{diversamente,} \end{cases}$$

dove $\vec{n_w}$ è il vettore normale alla parete, d_s la distanza di sicurezza a cui il pedone preferisce tenersi rispetto ai muri, r_i il raggio del pedone rappresentato come un disco, d_{iw} la distanza minima del pedone dalla parete e κ una costante che indica la *steepness*, cioè la ripidezza, della forza repulsiva.

Ovviamente la forza totale che agisce sul pedone è data dalla sommatoria di tutti i contribuiti dati dalle singole pareti:

$$\vec{F} = \sum \vec{F_w}, \vec{F_w} \in \mathcal{W}$$

3.1.4 Avoidance dei pedoni

Si definisce una distanza psicofisica ρ_i, alla quale il pedone preferisce tenersi dagli altri per sentirsi a proprio agio, tale che $\rho_i > r_i$, che definisce uno spazio discoidale $\mathcal{B}(x_i, \rho_i)$ centrato nella posizione attuale del pedone e di raggio ρ_i, che ne delimita lo spazio personale. Come si è detto nella prima parte di questo capitolo, questo modello di pedone cerca di ricavare dall'ambiente circostante informazioni sul comportamento degli altri, per evitare collisioni nell'immediato futuro. La condizione che indica una collisione ad un istante $t_c \geq 0$ si verifica quando un pedone P_j invade lo spazio personale di un altro pedone P_i, cioè quando:

$$\exists t_c \geq 0 | d_{ij} \leq \rho_i + r_j,$$

dove $d_{ij} = \|x_i - x_j\|$ è la distanza fra i centri dei due pedoni. Il pedone è in grado di risolvere potenziali collisioni che avvengono entro un certo tempo t_α di anticipo modificando la propria traiettoria. Per simulare questo comportamento si applica al pedone P_i una forza evasiva $\vec{F_e}$. Questa è la novità principale del modello scelto. L'algoritmo di avoidance consiste in quattro step:

- previsione delle collisioni;
- scelta dei pedoni;
- manovre di avoidance;
- calcolo della forza evasiva;

Previsione delle collisioni

Come prima cosa viene calcolato il set di pedoni $CP_i^{t_\alpha}$ che, dato un certo tempo t_α, avranno collisioni con il pedone P_i in esame. Per fare ciò, si calcola prima di tutto la velocità desiderata $\vec{v_i}^{des}$ del pedone P_i, come la somma della sua velocità attuale e di quella che deriverebbe dall'applicazione delle sole forze di attrazione del target e repulsive dei muri:

$$\vec{v_i}^{des} = \vec{v_i} + \left(\sum \vec{F_w} + \vec{F_g} \right) \Delta t,$$

dove Δt rappresenta la durata dello step della simulazione. A questo punto si calcola la futura posizione di P_i in base a quella attuale e alla velocità appena calcolata:

$$x_i' = x_i + t\vec{v_i}^{des}$$

Allo stesso modo le posizioni future degli altri pedoni che P_i può vedere sono calcolate estrapolando la loro attuale traiettoria: il pedone può solo stimare la loro velocità attuale non conoscendo il loro target. La posizione futura di un pedone P_j vista da P_i dunque è:

$$x_j' = x_j + t\vec{v_j}$$

A questo punto è possibile verificare se vi sarà una collisione fra due pedoni. Come definito precedentemente, questa avverrà se in un determinato istante si verificherà un'intersezione fra il disco di centro x_i e raggio ρ_i e il disco di centro x_j e raggio r_j, problema sintetizzato dall'equazione:

$$\| x_j - \left(x_i + \left(\vec{v_i}^{des} - \vec{v_j} \right) t \right) \| = \rho_i + r_j$$

Risolvendola per t è possibile verificare se esiste una possibile collisione futura. Se l'equazione non ha soluzioni o ne ha una sola, non vi saranno collisioni; se invece vi sono due soluzioni, che definiamo $t1$ e $t2$, vi sono tre distinte possibilità:

- $t1, t2 \leq 0$: si tratterebbe di una collisione passata, può quindi essere trascurata.
- $t1 < 0 < t2 \vee t2 < 0 < t1$: si tratta di una collisione imminente, per cui il pedone P_i viene inserito nel set $CP_i^{t_\alpha}$.
- $t1, t2 \geq 0$: una collisione avverrà nel tempo $tc_{ij} = min(t1, t2)$. Se $tc_{ij} \leq t_\alpha$ il pedone P_i viene inserito nel set $CP_i^{t_\alpha}$.

Scelta dei pedoni

Una volta calcolato il set $CP_i^{t_\alpha}$, esso viene ordinato in base al tempo di collisione e si considerano i primi N pedoni. Secondo gli autori del modello si ha già un buon comportamento fluido per valori piuttosto bassi di N, compresi fra 2 e 5. Questo permette di contenere i tempi di calcolo e allo stesso tempo di avere un maggiore realismo: anche nella realtà il singolo individuo riesce ad evitare un numero limitato di altri pedoni, solitamente quelli con cui potrebbero avvenire collisioni nei tempi più brevi.

Manovre di evasione

In base alle previsioni delle future posizioni dei pedoni P_i e P_j, che definiamo con $c_i = x_i + tc_{ij}\vec{v_i}^{des}$ e c_j (il calcolo è analogo), è possibile calcolare la forza evasiva che permetterà a P_i di evitare la collisione con P_j. La direzione è data dal versore $\vec{n}_{c_i c_j} = \frac{c_i - c_j}{\|c_i - c_j\|}$, nella direzione da c_j a c_i. Il valore della forza di evasione $\vec{F_{ij}}$ è approssimato da una funzione a tratti f(D), dove $D = \|\vec{c_i} - x_i\| + (\|\vec{c_i} - \vec{c_j} - r_i - r_j)$, cioè la distanza fra la posizione attuale del pedone P_i e la sua futura posizione $\vec{c_i}$ sommata alla distanza fra i due pedoni al momento della collisione. La funzione è composta da quattro tratti in quattro intervalli delimitati dalle soglie d_{min} (che definisce una barriera impenetrabile fra i pedoni), d_{mid}, che segna l'inizio di un tratto costante, d_{max} entro la quale la manovra evasiva avviene.

Calcolo della forza evasiva

Nell'ultimo passo dell'algoritmo viene calcolata la forza evasiva totale. Vi sono due approcci possibili: il primo, più semplice, consiste nel calcolo della forza esercitata da ciascun pedone appartenente al set $CP_i^{t_\alpha}$ e poi nella somma di queste secondo fattori di peso differenti; la seconda modalità è molto differente e consiste nell'applicazione iterativa delle singole forze e nella verifica a ogni step delle future collisioni.

3.2 Implementazione in Alchemist

Come già accennato, per l'implementazione in Alchemist del nuovo modello di pedone si è tenuto particolarmente conto della possibilità di estendere in futuro questo elemento. Si è pertanto proceduto alla modellazione tramite una classe generica e astratta *PredictivePedestrian*, che implementa i calcoli fin qui descritti e una classe *SAPERE-PredictivePedestrian* che ne rappresenta l'estensione nell'incarnazione SAPERE. Unico elemento esterno, che è necessario modellare in una classe specifica che estenda la prima, è la ricerca del target: come prima prova è stato implementato un pedone definito *VisualGoalFollowingPredictivePedestrian*, che come suggerisce il nome ricerca un obiettivo all'interno del proprio raggio visuale. Utilizzando questa classe è stato già possibile produrre alcune semplici simulazioni, come ad esempio quella del comportamento di due gruppi di pedoni nei pressi di un incrocio, in totale assenza di nodi di infrastruttura. Non è ovviamente possibile, in questa implementazione, uno steering dei pedoni. Una volta verificato il funzionamento di questo semplice modello, è stato inizialmente potenziato creando un oggetto *InfrastructureNode* che rappresenta appunto il nodo dell'infrastruttura, ancora da ricercare a livello visivo. Tramite esso è possibile avere un primo modello di gradiente, simile a quello finale. Si è successivamente implementata la versione finale del pedone, *GradientFollowingPredictivePedestrian*, la quale ricerca come primo obietti-

vo appunto un gradiente. Per avere un maggiore realismo, il pedone modellato tende a seguire, nel caso in cui non riesca a trovare un gradiente nell'ambiente intorno a sé, il pedone più vicino. I valori delle costanti non esplicitate dagli autori del modello sono stati trovati per via sperimentale ed è possibile modificarli per modellare comportamenti differenti. Si è inoltre deciso di rendere disponibile esternamente il livello di attenzione del pedone, cioè il numero di altri individui che è in grado di evitare: sono stati scelti tre livelli, il primo dei quali considera un solo pedone, il secondo tre e il terzo cinque. Come indicato dagli autori, questo numero è più che sufficiente per avere un comportamento realistico. Si è tuttavia deciso, ai fini sperimentali, di dare la possibilità di scegliere un numero arbitrario di pedoni da tenere in considerazione.

3.3 Simulazioni generate e test effettuati

La prima simulazione, come detto, rappresenta un incrocio con due flussi di pedoni che si incontrano; scopo principale è testare il funzionamento del modello, controllare che non si verifichino collisioni e avere un'idea del livello di realismo. Il comportamento dei pedoni risulta piuttosto naturale, il movimento fluido ed effettivamente non vi sono criticità o scontri. La figura 3.1 mostra alcuni fotogrammi di questa simulazione, la cui particolarità è l'indipendenza dai nodi di infrastruttura: l'inseguimento del target (che per ogni gruppo di pedoni si trova dalla parte opporta rispetto all'incrocio) avviene esclusivamente in maniera visuale. Grazie a questa semplice simulazione è stato inoltre possibile mostrare la differenza fra i tre livelli di attenzione: come mostra la figura 3.2, il pedone che considera solamente un altro individuo risulta più rallentato, pur mantenendo la capacità di prevedere ed evitare collisioni. Non si evidenziano, invece, particolari differenze fra i due livelli superiori (saranno più accentuate, anche se non fondamentali, in simulazioni più complesse). Evoluzione di questa è la simulazione di un grande ambiente con un buon numero di pedoni e pochi nodi di infrastruttura, che saranno gradualmente aumentati fino ad avere una copertura piuttosto fitta dell'ambiente. Si sceglie quindi una pianta di un edificio da utilizzare per i test, i cui risultati saranno esposti nel capitolo apposito.

3.4 Limiti

A differenza del modello precedente, questo risulta migliore nell'evitare gli altri pedoni in movimento, dato che non si basa sulla loro posizione attuale, ma cerca di intuire quale essa sarà negli istanti successivi. Altro punto di forza molto importante è dato dalla ricerca degli altri pedoni, che è visiva e non più tramite il *neighborhood* dato dalla linking rule. Questo alleggerisce il calcolo e rende il modello più realistico, dato che

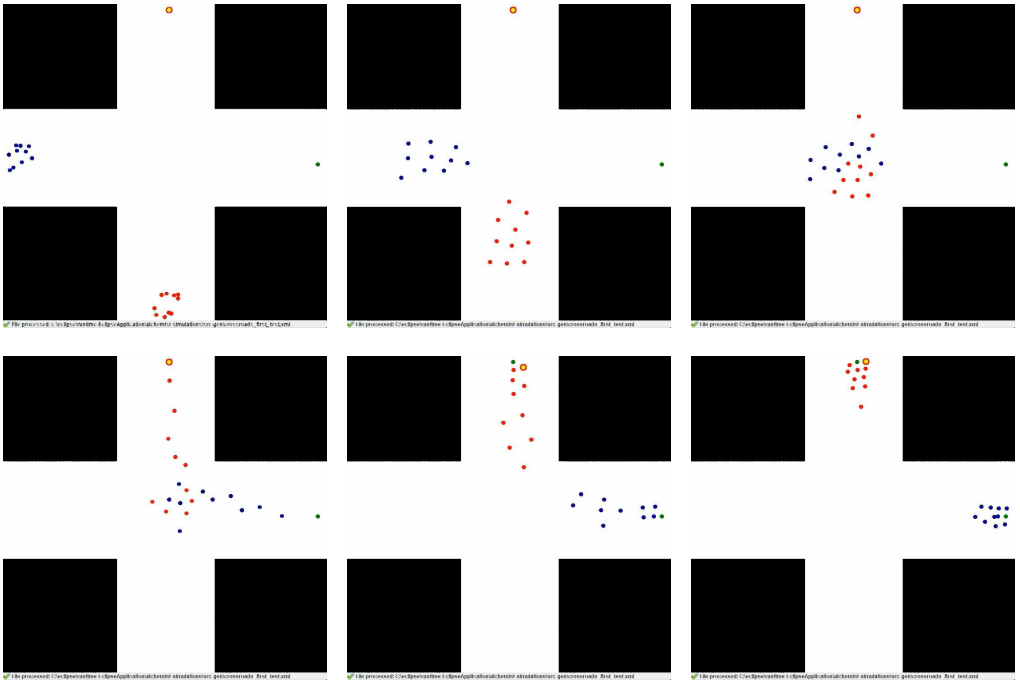

Figura 3.1: *Alcuni fotogrammi della simulazione dell'incrocio.*

diversamente un pedone potrebbe non accorgersi della presenza di un altro anche molto vicino se non fossero collegati allo stesso nodo di infrastruttura. Il limite più significativo al momento è dato dalla capacità di riconoscere gli ostacoli da tenere in conto nel calcolo delle forze: la modellazione dell'ambiente (ricavato, come già detto, da un'immagine che viene scomposta in rettangoli) può generare molti ostacoli di dimensioni minime in un raggio piuttosto limitato, ed è molto difficile allo stato attuale delle cose discriminare quali di questi siano veramente visibili e quali no. Allo stesso modo, il fatto che il pedone sia approssimato da un disco mentre il sistema tratta i nodi come punti materiali ha costretto a complicare i calcoli per evitare che il pedone si trovi a una distanza troppo breve da una parete.

Figura 3.2: *I tre livelli di attenzione a confronto: a sx il liv. 3, al centro il liv. 2, a dx il liv. 1*

4 Risultati

In questo capitolo verranno presentati i test effettuati sul nuovo modello di pedone e i risultati ottenuti.

4.1 Simulazione di partenza

Per il test dei risultati è stata realizzata una simulazione base utilizzando come ambiente una pianta di un edificio (si tratta in particolare della Basilica di S. Pietro in Vaticano), come linking rule la *SelectiveEuclideanDistanceWithObstacles*, cioè una regola che mette in relazione con i nodi di una specifica tipologia (nella fattispecie, i nodi di infrastruttura), compresi entro un certo raggio (definito a 20 metri) e con un limite massimo di link per nodo, che in questo caso non è stato specificato. Nell'ambiente sono stati inseriti 5 POI, di cui i primi tre si trovano sugli ingressi della Basilica e gli ultimi due alle opposte estremità del transetto. Sono poi stati aggiunti 117 nodi di infrastruttura, a distanza regolare di 10 metri l'uno dall'altro, per creare una griglia di collegamento, contenenti la reazione *SAPEREGradient* e l'action *CrowdSensor* già citate in precedenza. In ultimo sono stati posizionati 400 nodi contenenti l'agente *GradientFollowingPredictivePedestrian* appena realizzato, con il nuovo modello di pedone, impostando i parametri come descritto:

Molecola da inseguire il gradiente sopra definito

Livello di attenzione 3

Range di ricerca 20 metri

Il rate della reazione è stato settato a 1 per avere una simulazione con buoni risultati in real-time. Il codice completo della simulazione base è riportato in appendice. Nella simulazione (la cui situazione iniziale è visibile in figura 4.1) sono stati indicati in blu i pedoni in movimento, in verde i nodi dell'infrastruttura e in rosso i nodi target.

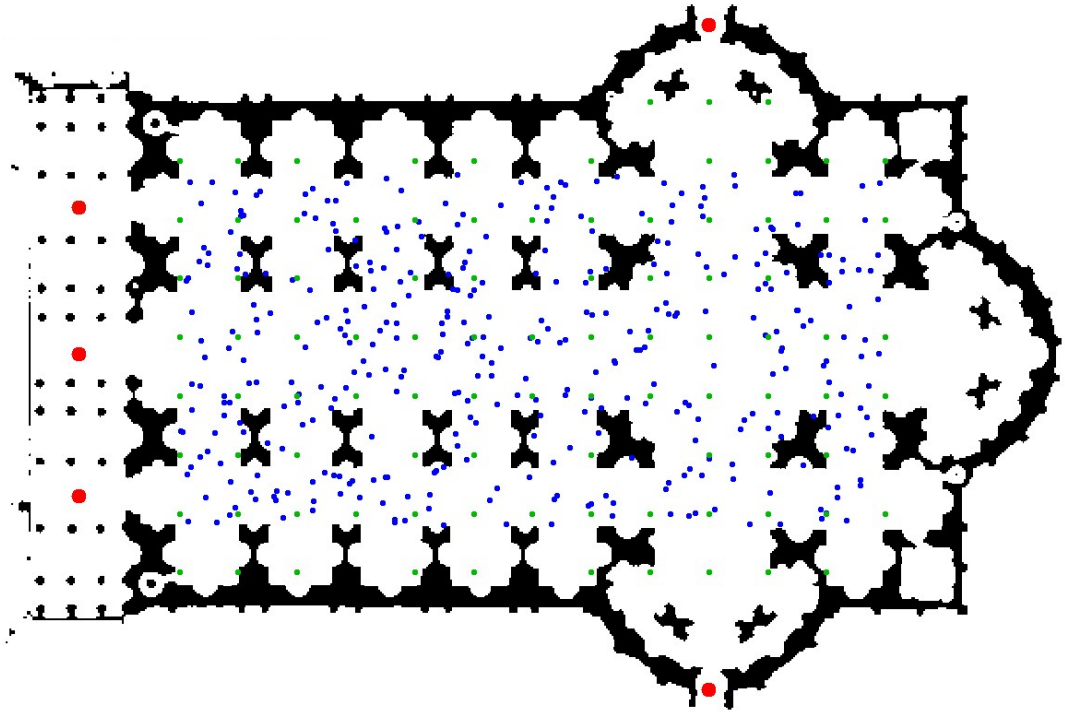

Figura 4.1: *La simulazione standard*

4.2 Parametri di test

Lanciando la simulazione originale, si è verificato il tempo necessario affinché tutti i pedoni simulati raggiungessero uno dei POI. Lo stesso test è stato ripetuto variando singolarmente ciascuno dei parametri indicati di seguito. Sono state realizzate diverse simulazioni modificando uno alla volta questi parametri e andando a confrontare il tempo necessario a completare la simulazione rispetto alla versione base. Nota: a seconda del tipo di test effettuato è stato necessario - per esigenze di tempo da una parte e di precisione dall'altra - variare la reattività della simulazione, per cui la stessa versione originale può avere tempistiche diverse a seconda del test. Questo significa che al variare dello stesso parametro tutte le simulazioni sono confrontabili, ma lo stesso non è possibile su casi differenti.

- Livello di attenzione
- Numero di pedoni
- Livello di dettaglio dell'immagine rappresentante l'ambiente
- Numero di nodi di infrastruttura e loro distanza

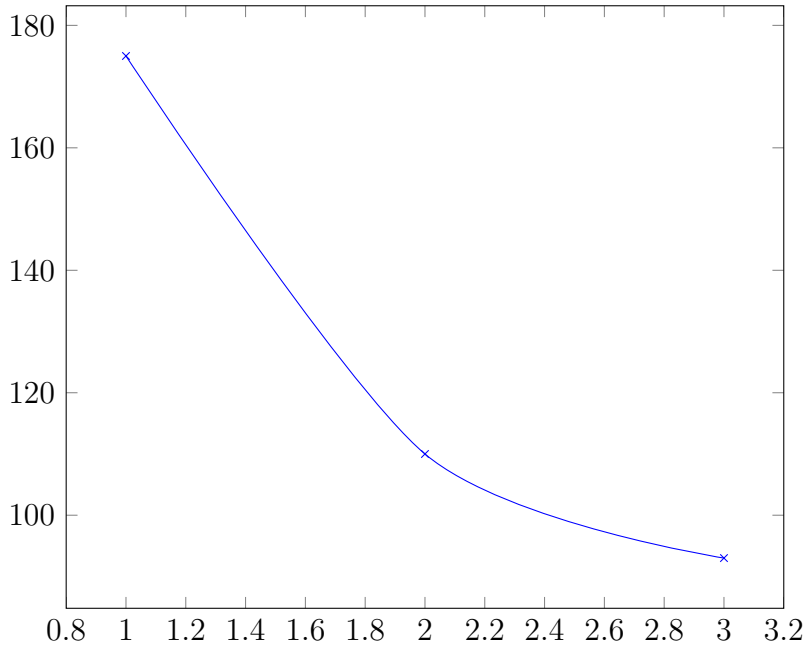

Figura 4.2: *Tempo di simulazione al variare del livello di attenzione*

- Rate della reazione

4.2.1 Livello di attenzione

La simulazione è stata lanciata usando i tre diversi livelli di attenzione possibili, che identificano il numero di pedoni da considerare nel calcolo delle forze di avoidance:

1: un solo pedone

2: 3 pedoni

3: 5 pedoni

Il risultato ottenuto è sintetizzato dal grafico in figura 4.2: sembra che a un livello di attenzione più alto corrisponda un vantaggio anche piuttosto forte (dal livello 1 al 3 il tempo di simulazione si dimezza quasi), derivato dal fatto che il pedone più attento evita un numero maggiore di possibili collisioni. Il risultato è dovuto inoltre alla complessità della simulazione, mentre si era verificato che in casi molto più semplici il livello che garantiva risultati migliori fosse il 2, con un buon compromesso fra prestazioni e risultato.

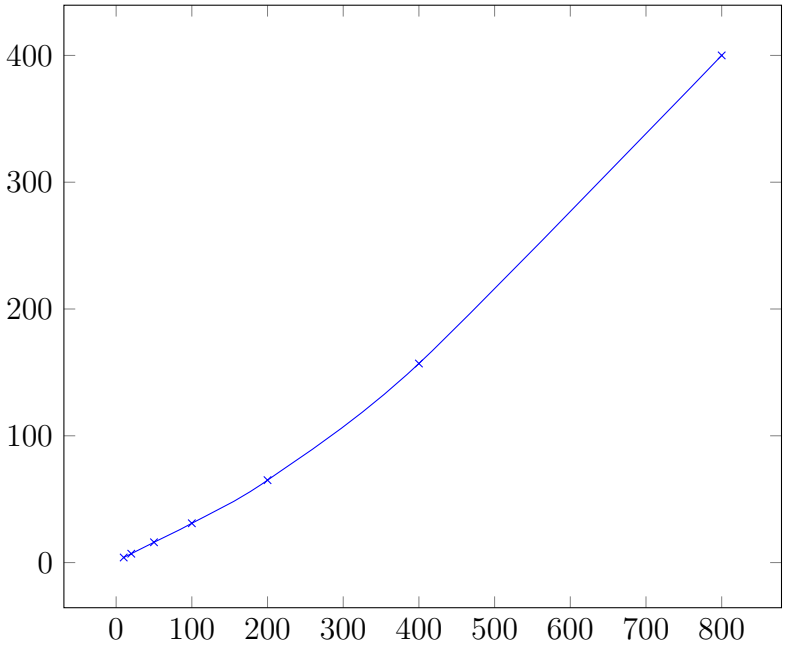

Figura 4.3: *Tempo di simulazione al variare del numero di pedoni*

4.2.2 Numero di pedoni

La modifica del numero di pedoni, come mostra il grafico (figura 4.3), può incidere in maniera significativa sulle prestazioni del sistema. Risulta del resto facilmente intuibile che a un numero elevato di pedoni corrisponda una quantità sempre più alta di computazioni da svolgere nell'arco di uno step di simulazione. Per valori molto alti risulta quindi impossibile ottenere un risultato real-time, in quanto per avere un secondo di simulazione è necessario un tempo di calcolo superiore. Va comunque precisato che le prestazioni restano buone, aumentando la reattività del sistema, anche per valori molto più alti.

4.2.3 Livello di dettaglio dell'immagine

Un elemento che può incidere notevolmente sulle prestazioni del sistema è costituito dal livello di dettaglio e dalla qualità dell'immagine utilizzata come ambiente di simulazione: infatti al caricamento la figura viene scomposta in una serie di rettangoli, per cui in presenza di linee curve o frastagliate verrà creato un numero elevato di piccoli ostacoli. Questo ha un notevole impatto nel calcolo della forza di avoidance dalle pareti. Al fine del calcolo delle prestazioni, dalla stessa immagine sono state create 4 copie abbassando per ognuna la qualità utilizzando la tecnica più semplice possibile, ovvero quella del

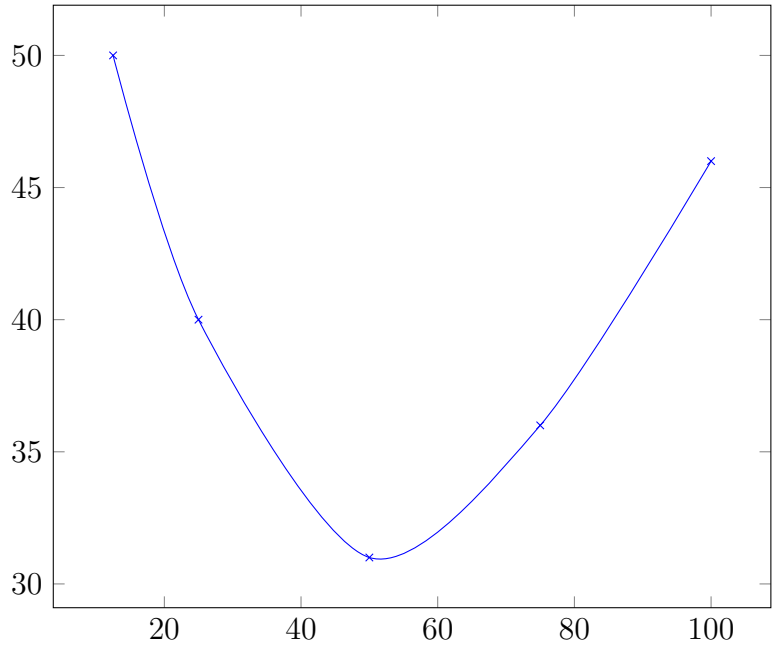

Figura 4.4: *Tempo di simulazione al variare della definizione dell'ambiente (qualità dell'immagine)*

downscaling/upscaling: l'immagine viene ridimensionata utilizzando un fattore di scala via via più alto e poi riportata alle dimensioni originali, perdendo nel processo un certo livello di qualità. Sono stati quindi considerati 5 livelli di definizione dell'immagine, in base alla rapporto fra dimensione scalata e originale, a partire dal 100% (qualità massima) fino al 12.5%. La figura 4.4 mostra il risultato ottenuto: come si può notare dal grafico, la velocità della simulazione aumenta al calare della qualità dell'immagine, ma oltre un certo livello (costituito dal 50% circa) ha di nuovo un peggioramento. Questo si può spiegare andando a confrontare l'immagine originale (fig. 4.5) con quella a qualità ridotta a 1/8, riportata in fig. 4.6. Si evidenzia come abbassando la qualità dell'immagine oltre un certo livello si possano produrre artefatti o lacune che vanno a modificare completamente l'ambiente originale, che si trova ad avere aperture o ostacoli non presenti nell'originale. Si conclude dunque che il migliore rapporto tra velocità di simulazione e qualità dell'immagine si ha quando questa è ridotta al 50% ma non oltre.

4.2.4 Numero di nodi di infrastruttura e distanza

Anche la variazione del numero di nodi dell'infrastruttura ha il suo peso nella velocità della simulazione. Per determinare quanto forte sia questa influenza si è partiti da un valore minimo di 12 nodi, disposti manualmente in modo da coprire tutto l'ambiente

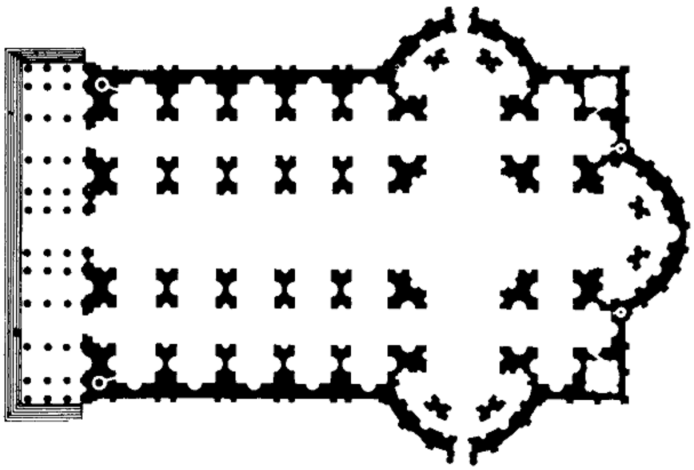

Figura 4.5: *L'ambiente di simulazione standard*

(ovviamente tarando opportunamente il range della linking rule e della ricerca dei pedoni), per poi aumentare il numero di elementi dell'infrastruttura, disponendoli a intervalli regolari e abbassando il valore del range, fino a circa 400 punti a distanza di 5 metri l'uno dall'altro. L'andamento del grafico (figura 4.7) suggerisce, ove possibile, di tenere un numero di elementi di infrastruttura tale da garantire la copertura dell'ambiente ma senza sovraccaricarlo. Ovviamente in alcuni casi questo numero potrebbe essere vincolato, e occorrerebbe agire opportunamente sugli altri parametri in gioco, in primis il range di ricerca dei pedoni e quello della linking rule.

4.2.5 Rate della reazione

Il rate della reazione costituisce il numero di esecuzioni nell'unità di tempo. Più il valore è alto, quindi, più la simulazione risulterà fluida e gradevole alla vista, mentre valori bassi tenderanno a produrre andamenti a scatti. Per contro, ovviamente, aumentare questo parametro causerà un maggiore carico sulla CPU, andando a rallentare la velocità di simulazione. Per quanto, come già detto, i risultati (mostrati nel grafico in figura 4.8) non siano da considerare in senso assoluto, si nota immediatamente che a un valore pure non troppo alto del rate (8) il tempo di esecuzione è già triplicato. Mentre nel caso di simulazioni semplici è possibile tenere valori del rate anche molto alti, quando l'ambiente ha una certa complessità e il numero di nodi (pedoni e di infrastruttura) è sufficientemente alto occorre che questo valore rimanga entro un certo limite. Nel caso

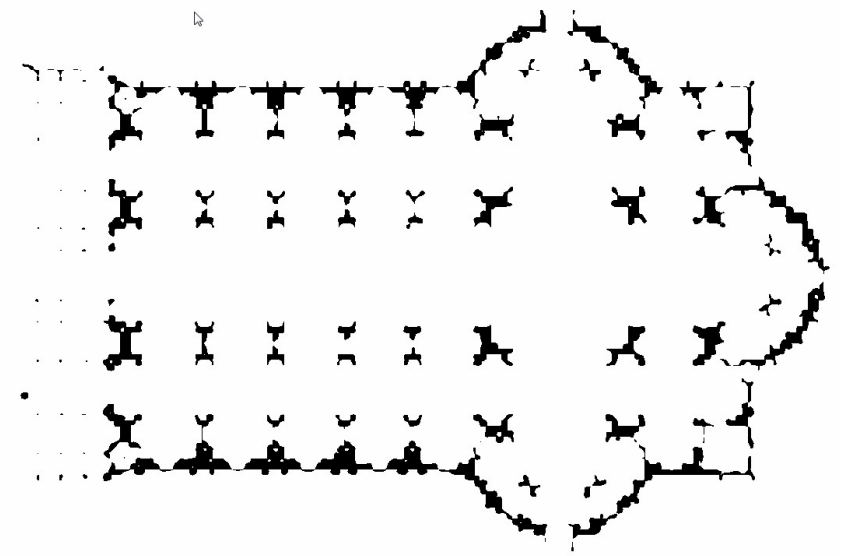

Figura 4.6: *L'ambiente di simulazione a qualità 12.5%*

in cui la simulazione non riesca ad avvenire in realtime sarà comunque possibile tramite il cronometro dell'interfaccia grafica determinare quale sia il valore reale.

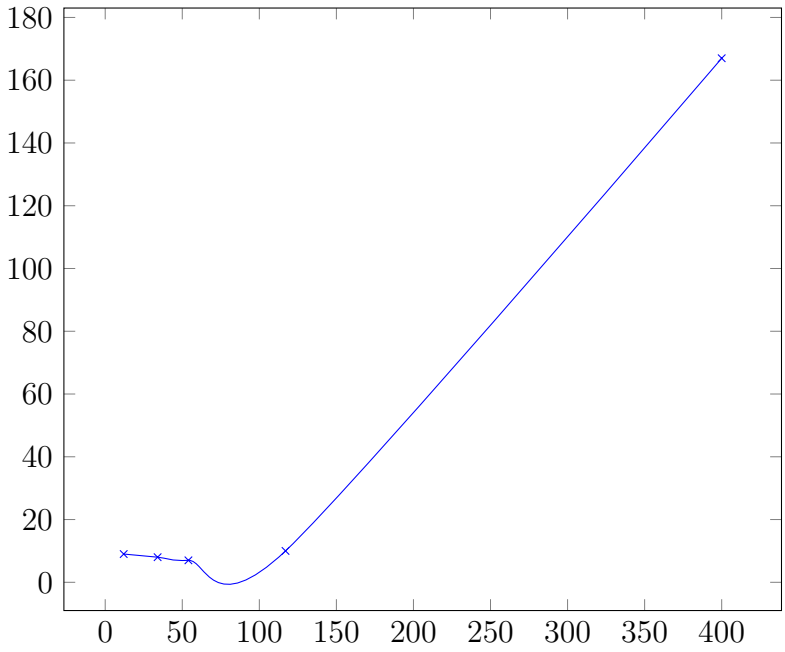

Figura 4.7: *Tempo di simulazione al variare del numero di nodi di infrastruttura*

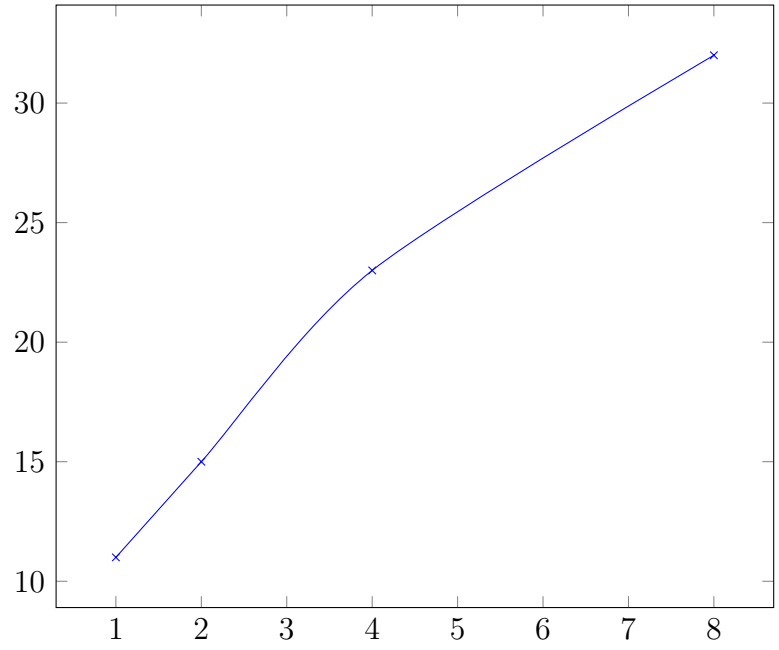

Figura 4.8: *Tempo di simulazione al variare del rate della reazione*

5 Algoritmi di crowd steering

In questo capitolo saranno analizzate le simulazioni con cui è stato testato l'algoritmo di crowd sensitiveness sui due modelli di pedone, presentando i risultati ottenuti. La guida di un gruppo di pedoni è effettuata nelle simulazioni realizzate tramite un gradiente presente nei nodi di infrastruttura, che indica la distanza dal target più vicino. Questo valore può essere eventualmente modificato per permettere appunto lo steering. Nella realtà il tutto può essere realizzato tramite reti di sensori (parte di sensing) e monitor per indicare la direzione, oppure sfruttando la diffusione ormai capillare di dispositivi wireless è possibile utilizzare questi stessi dispositivi, opportunamente connessi a una rete di access point.

5.1 Crowd sensitiveness

L'algoritmo analizzato, detto *crowd sensitiveness*, si basa sulla percezione della quantità di pedoni presenti all'interno di una determinata area. In base a questa vengono modificati alcuni valori presenti nell'ambiente, che a loro volta sono letti dai pedoni e utilizzati per decidere la direzione da prendere.

5.2 Implementazione in Alchemist

In Alchemist la crowd sensitiveness è modellata in una classe *CrowdSensor* fra le actions disponibili all'interno del pacchetto del framework. Questa compie un'operazione molto semplice: cerca nel neighborhood del nodo in cui è contenuta (quindi fra i nodi a cui è collegato dalla linking rule) tutti quelli che contengono una molecola "person" e ne inserisce la quantità totale N all'interno di una molecola "crowd, N". Tra i parametri della reaction *SAPEREGradient* è poi possibile specificare questa molecola *crowd* come modificatore del valore del gradiente e la formula, che solitamente contiene fra i propri parametri la distanza dal target. Sono stati analizzati due casi, uno in cui il valore di *crowd* contribuisce a questa distanza sommandovi una penalità proporzionale al proprio valore, e uno in cui la modifica moltiplicandola per il proprio valore.

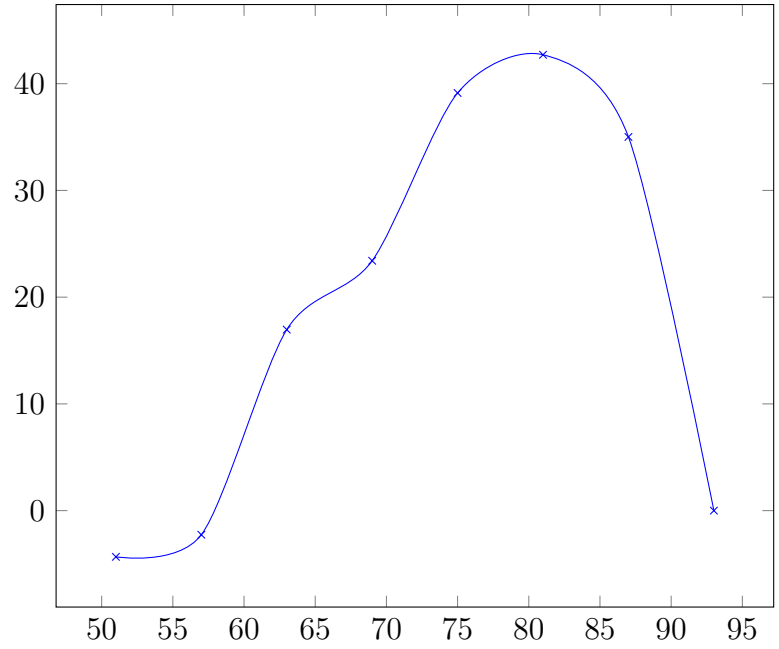

Figura 5.1: *Risultato al 50%: guadagno in tempo (in %) al variare del numero di pedoni*

5.3 Test effettuati sul modello SocialForce

Per avere una misurazione quantitativa della validità degli algoritmi di steering è stato utilizzato il tool di model checking precedentemente descritto per lanciare diverse migliaia di simulazioni e sono stati graficati i risultati. Come situazione di partenza, si ha una simulazione con questi elementi:

Environment l'ambiente della fiera precedentemente citato.

Infrastruttura 628 nodi posti in una griglia di passo 0.48 metri

Molecola da inseguire il gradiente generato da alcuni nodi posti a un'estremità dell'ambiente, con *CrowdSensor* con effetto additivo

Pedoni da guidare 7 nodi posti all'estremità opposta dell'ambiente

Pedoni ostacolo 57 nodi posti nei dintorni dei nodi target, a inseguimento di un gradiente che ha origine nell'intorno dei pedoni da guidare: in questo modo i percorsi dei due gruppi di pedoni si incrociano circa a metà dell'ambiente.

Sono state generate due versioni, una con steering abilitato e una senza guida dei pedoni, e per ognuna è stato realizzato un set di simulazioni, andando ad aumentare i pedoni

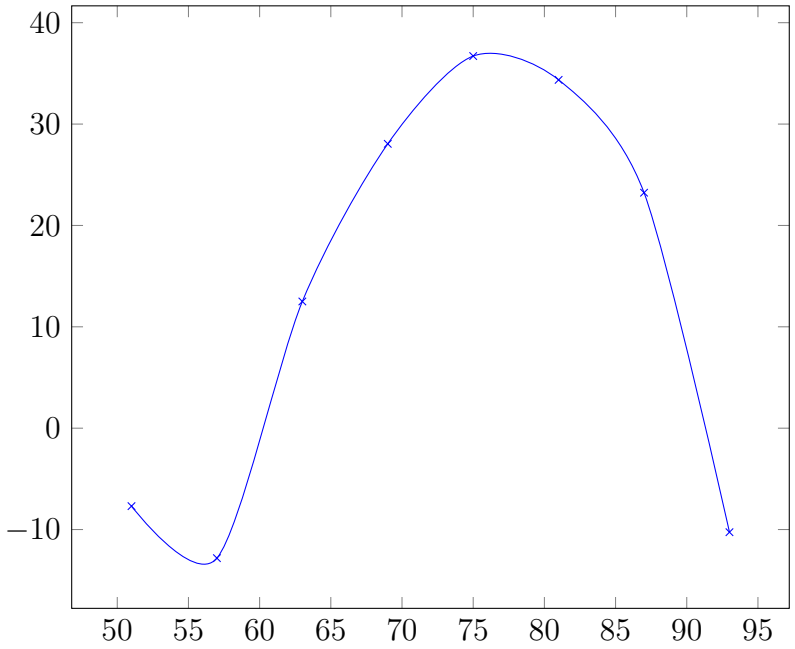

Figura 5.2: *Risultato al 75%: guadagno in tempo (in %) al variare del numero di pedoni*

ostacolo di 3 elementi a ogni step, fino ad un massimo di 93 (questo limite è stato trovato empiricamente e dipende dalle caratteristiche dell'ambiente), oltre il quale non vi è più un guadagno significativo. Sono stati analizzati tramite model checking i tempi in cui il 50%, 75% e 90% dei pedoni raggiungeva il target. Nei grafici che si trovano in queste pagine è rappresentato il guadagno al variare del numero di pedoni, inteso come il rapporto fra il tempo della simulazione con steering abilitato e quello della simulazione con algoritmo disabilitato.

5.3.1 Risultato al 50%

Come facilmente intuibile (figura 5.1), per valori bassi del numero di ostacoli il rallentamento sul pedone non guidato non è sufficiente a far sì che lo steering porti un vantaggio, vantaggio che invece è evidente con un numero intermedio di pedoni ostacolo. Quando invece questo numero supera una certa soglia, anche i pedoni guidati trovano difficoltà nel raggiungere il target e il guadagno torna nuovamente a valori prossimi allo zero. I valori riportati nel grafico ovviamente non sono da intendere in senso assoluto, ma vanno considerati in proporzione alle dimensioni dell'ambiente, alla quantità e alla posizione degli ostacoli, al numero e alla posizione dei nodi di infrastruttura.

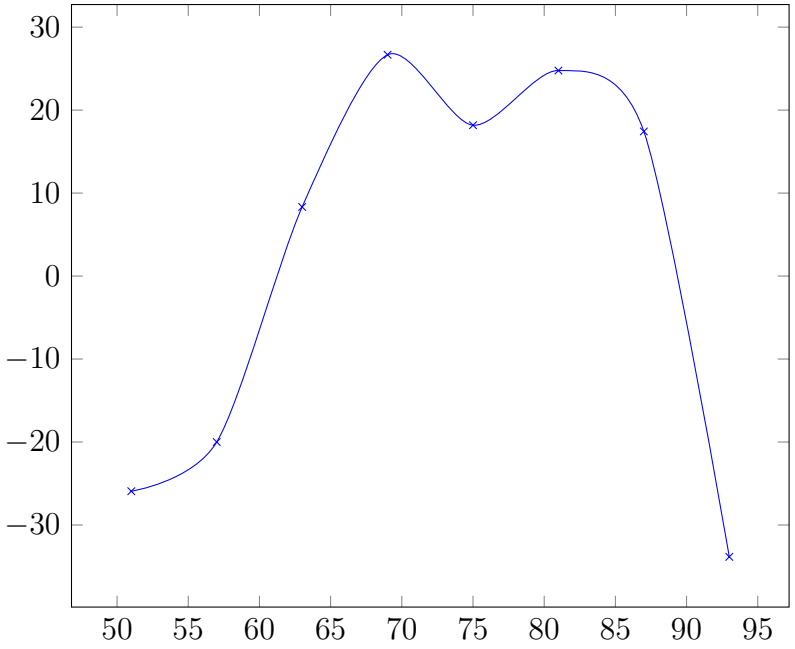

Figura 5.3: *Risultato al 90%: guadagno in tempo (in %) al variare del numero di pedoni*

5.3.2 Risultato al 75%

Come si vede in figura 5.2, prendendo in considerazione il 75% dei pedoni la forma della curva resta simile, mentre il guadagno massimo cala leggermente.

5.3.3 Risultato al 90%

Lo stesso avviene anche considerando il 90% dei pedoni (figura 5.3). Occorre tenere presente che quando il numero di persone è sufficientemente alto, man mano che si avvicinano al target vanno ad alterare il valore del gradiente, che quindi naturalmente tende a rallentare i pedoni ancora lontani dall'obiettivo. Per questo motivo il guadagno cala quando si prende in considerazione un maggior numero di nodi rispetto alla totalità.

5.4 Test effettuati sul modello predittivo

Il modello predittivo è stato invece utilizzato per testare la differenza fra il caso in cui il livello di affollamento influisce sul gradiente in maniera additiva e quello in cui agisce in maniera moltiplicativa. Sono inoltre state realizzate alcune simulazioni per dimostrare

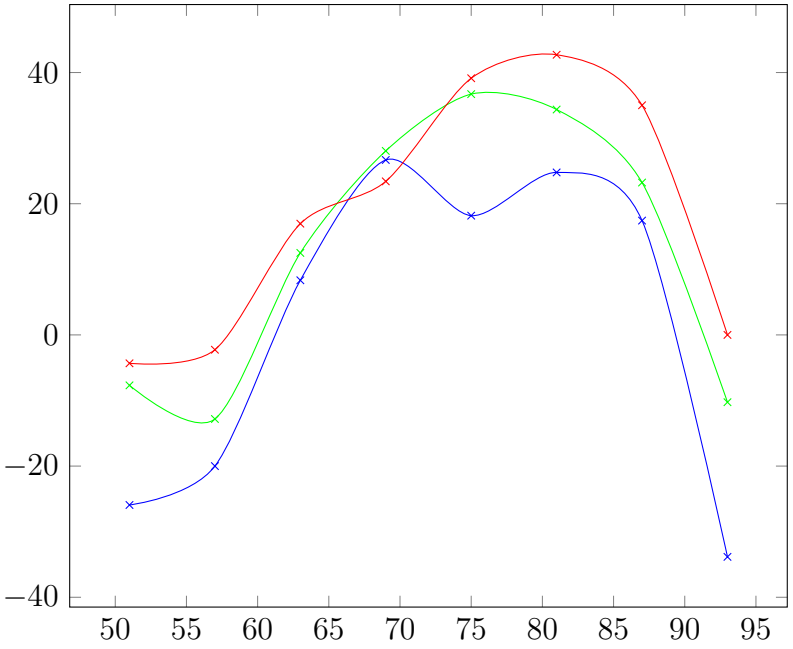

Figura 5.4: *Le tre curve a confronto*

come lo steering contribuisca, al di là dei risultati numerici, a suddividere un gruppo di pedoni fra più target. Questo è particolarmente utile per simulare casi di evacuazione, in cui si hanno ad esempio diverse vie di fuga e in cui, senza steering, tutte le persone tenderebbero a scegliere la più vicina. Per tutte le simulazioni è stata utilizzata come environment la pianta della basilica già vista in precedenza.

5.4.1 Evacuazione

Nel primo test è stato dunque modellato un ambiente con un numero maggiore di punti di interesse rispetto al solito, nella fattispecie 13, e una folla concentrata principalmente nel centro dell'ambiente. Obiettivo era verificare che questa si suddividesse fra i vari target andando a raggiungere anche i più lontani. Come mostra l'immagine 5.6, che mette a confronto lo stato iniziale e la condizione finale, vi è una buona distribuzione dei pedoni fra i punti di interesse. Per evidenziare maggiormente l'effetto si è realizzata una seconda simulazione con 5 soli punti di interesse, di cui tre agli ingressi e due alle estremità del transetto, come nella simulazione standard già utilizzata. Anche in questo caso, come mostra la figura 5.7, parte dei pedoni tende a dirigersi verso gli ultimi due obiettivi, nonostante la distanza sia molto maggiore.

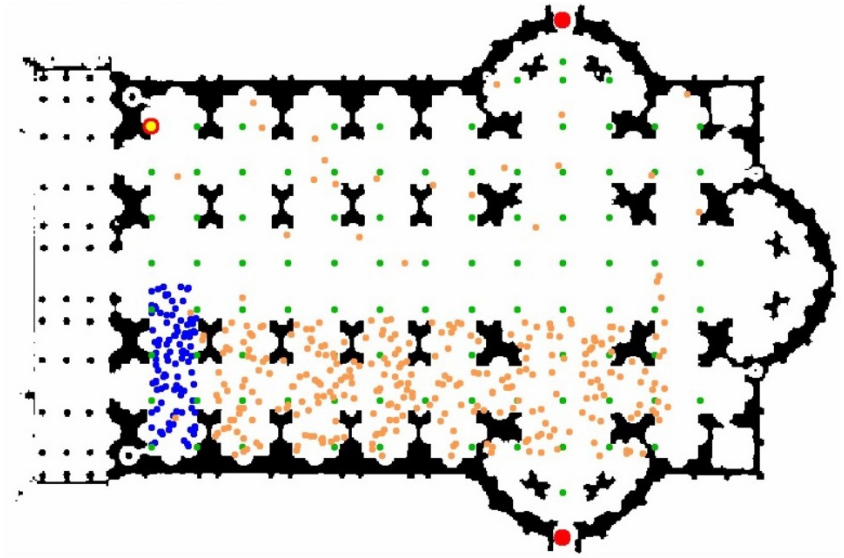

Figura 5.5: *La simulazione utilizzata per testare lo steering con il modello PredictivePedestrian*

5.4.2 Fattore additivo e moltiplicativo

Sono poi state generate alcune simulazioni per valutare come la decisione di utilizzare il numero di pedoni in senso additivo o moltiplicativo nella formula del gradiente influisca sul risultato finale. Si è quindi partiti da una condizione iniziale in cui vi è un certo numero di pedoni da guidare nella parte in basso a sinistra dell'ambiente, una quantità piuttosto alta di pedoni ostacolo (fermi) nella metà inferiore, alcuni sparsi nella parte superiore e due punti di interesse agli estremi del transetto, come rappresentato in figura 5.5. Di questo modello sono state create tre versioni, una con steering disabilitato, una con steering additivo e una terza con steering moltiplicativo. Confrontando i due estratti del codice delle simulazioni è evidente la differenza fra i due sistemi:

Listing 5.1: Pedoni

```
1 reaction SAPEREGradient params "ENV,NODE,RANDOM,source,gradient,2,((
    Distance+#D)+(2*L)),crowd,2000000,10" []-->[]
```

Listing 5.2: Pedoni

```
1 reaction SAPEREGradient params "ENV,NODE,RANDOM,source,gradient,2,(
    Distance+(#D*(0.5*L))),crowd,2000000,10" []-->[]
```

Nella versione additiva, come si nota, il valore di L va a sommarsi (moltiplicato per un coefficiente) al valore della distanza, mentre in quella moltiplicativa va appunto a moltiplicarsi. Per ognuna delle tre versioni, a sua volta, sono state generate diverse

simulazioni variando il numero dei pedoni ostacolo. Confrontando i risultati si possono fare le seguenti osservazioni:

- rispetto al modello precedente, per il pedone predittivo il guadagno ottenuto utilizzando lo steering è inferiore, in alcuni casi anche piuttosto significativamente; occorre però tenere presente che l'ambiente di simulazione ha dimensioni molto maggiori e che il modello stesso è più efficiente, per cui tende a comportarsi meglio in presenza di ostacoli, anche nel caso senza steering;

- fra i due tipi di steering, all'aumento del numero di ostacoli nell'ambiente, in particolare di quelli che non fanno parte del blocco da evitare, il guadagno della versione additiva cala, mentre quello della versione moltiplicativa, che inizialmente era più basso dell'altro, resta pressoché invariato. Questa seconda possibilità sembra quindi più robusta rispetto alle variazioni dei parametri della simulazione;

- per quanto il guadagno in tempo sia una misura interessante, non costituisce necessariamente l'unico parametro da tenere in considerazione nella valutazione dell'efficienza di un algoritmo di steering: considerando anche casi reali, a meno di non trovarsi in ambienti con un livello molto elevato di affollamento difficilmente un percorso più lungo darà vantaggi in termini di tempo. Resta però invariata l'importanza di avere una migliore distribuzione dei pedoni fra più target, importanza che diventa cruciale quando si vuole valutare la safety dei pedoni in casi di pericolo.

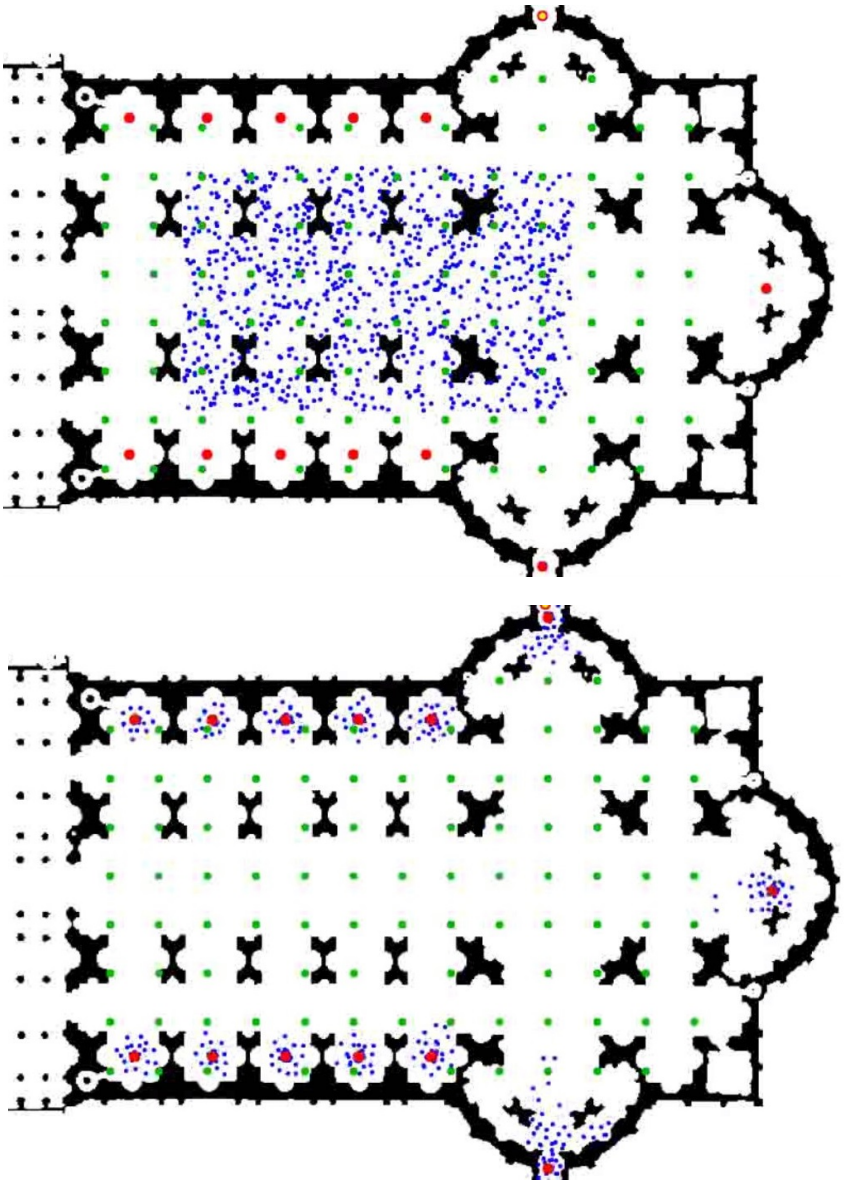

Figura 5.6: *Distribuzione dei pedoni fra 13 punti di interesse*

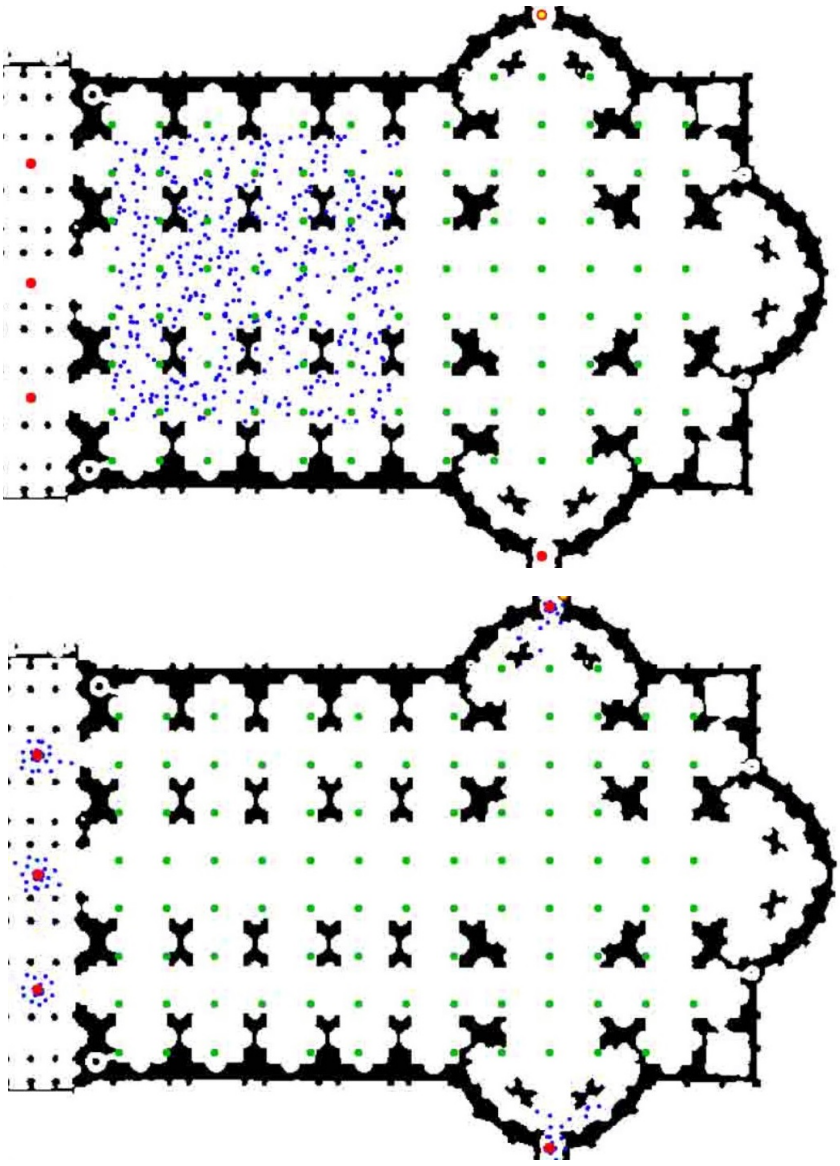

Figura 5.7: *Distribuzione dei pedoni fra 5 punti di interesse*

6 Conclusioni e sviluppi futuri

In quest'ultimo capitolo saranno tratte le conclusioni sul lavoro svolto e proposte alcune possibili evoluzioni per migliorare il sistema di simulazione e i modelli utilizzati.

6.1 Conclusioni

Al termine di questo lavoro sono stati raggiunti i seguenti obiettivi, raggruppati nei tre elementi fondamentali citati nell'introduzione:

Steering
: Gli algoritmi analizzati hanno dato una buona risposta in termini sia qualitativi che quantitativi: i due differenti set di simulazioni hanno evidenziato da un lato la possibilità di ottenere un buon guadagno in tempo, dall'altro quella di suddividere i pedoni fra più target in maniera realistica e fattibile.

Modello di pedone
: Durante i test si è prospettata la necessità di implementare un modello di pedone differente da quello già disponibile, modello che rispetto al precedente è risultato più indipendente dai nodi di infrastruttura, più realistico e leggero in termini computazionali

Simulazioni
: Le simulazioni prodotte e i risultati ottenuti costituiscono una buona base per dimostrare le prestazioni di Alchemist e un punto di partenza per la realizzazione di nuovi scenari.

6.2 Possibili sviluppi futuri

Saranno ora proposti alcuni possibili sviluppi per estendere il sistema di simulazione, o migliorarne le performance.

6.2.1 Framework

Le prime proposte riguardano il framework di simulazione, e in particolare la gestione degli ambienti:

- L'attuale caricamento di ambienti da file PNG, per quanto intuitivo e potente, in presenza di immagini piuttosto complesse tende a creare ambienti con una quantità molto elevata di ostacoli di piccole dimensioni. Ciò avviene particolarmente quando l'immagine è ricca di linee curve o frastagliate. Il forte aumento del numero di ostacoli rallenta notevolmente la simulazione, incidendo sulle performance, e rende inoltre molto più complesso per il modello del pedone il calcolo delle forze. Si potrebbe pertanto realizzare un environment semplificato, che mostri la mappa così com'è nell'immagine ma in cui per il calcolo delle forze vengano utilizzate delle approssimazioni, cercando di sommare in un unico rettangolo il maggior numero possibile di ostacoli, oppure dando la possibilità di modellare gli ostacoli come poligoni invece di pedoni.

- Allo stesso modo potrebbe essere interessante l'utilizzo di piante di interni in formato vettoriale, come già è possibile fare con mappe in formato OpenStreetMaps. Questo dovrebbe semplificare notevolmente calcoli e simulazione, oltre a fornire un buon supporto per modellare casi realistici di evacuation.

6.2.2 Modellazione dei pedoni e algoritmi di steering

Seguono alcune idee per migliorare la modellazione dei pedoni e proposte su nuovi algoritmi di steering:

- A livello di framework sarebbe molto utile avere una classe di supporto che permetta di utilizzare nelle simulazioni dischi di un determinato raggio. Il sistema infatti considera i nodi come punti materiali, e di conseguenza la verifica che questi non si trovino troppo vicini a una parete o a un altro pedone (caso nella realtà impossibile, dato che ogni persona occupa comunque una determinata area) è demandata al codice del pedone stesso, rendendolo più lento e pesante a causa delle verifiche che occorre fare a ogni ciclo in seguito al calcolo delle forze. Al momento l'ambiente permette solamente di calcolare la prossima posizione consentita in una direzione, fino al bordo di un ostacolo. Occorrerebbe invece tenere in considerazione appunto il raggio del disco che modella il pedone.

- Finora sono stati presi in considerazione modelli di pedoni in condizioni normali, mentre in caso di pericolo, forte stress o panico il comportamento è notevolmente differente. Sarebbe dunque molto utile modellare un pedone che si muova in maniera più realistica in caso di condizioni estreme.

- Il modello di pedone implementato durante lo svolgimento di questa tesi può essere modificato rendendo disponibili all'esterno un numero maggiore di parametri (raggio, velocità preferita, distanza preferita dalle pareti, ecc.), permettendo di modellare anche le differenze comportamentali che vi sono naturalmente fra persone di età, sesso, cultura diverse.

- Steering predittivo: se è il sistema stesso a indicare ai pedoni la direzione in cui muoversi, esso può prevedere con una buona approssimazione quali saranno le prossime posizioni, e migliorare di conseguenza gli algoritmi stessi di guida.

6.2.3 User experience e interfaccia

- La GUI di Alchemist è stata notevolmente potenziata nel tempo, aggiungendo la possibilità di scegliere tra soluzioni di output differenti. Per semplificare la realizzazione di simulazioni si potrebbe riversare direttamente su file il contenuto della finestra di output, realizzandone quindi direttamente il filmato.

- Si potrebbe inoltre riprendere e migliorare l'integrazione con Blender, già realizzata a suo tempo da chi scrive: in output vengono scritte su file di testo le posizioni di pedoni e ostacoli, che è quindi possibile importare in Blender con gli appositi script per avere un rendering tridimensionale dell'ambiente. Integrando questo con librerie che permettano la modellazione 3D di figure umane (ve ne sono diverse anche di libero utilizzo) si possono raggiungere livelli di realismo molto forti.

Bibliografia

[1] Jose Luis Fernandez-Marquez, Giovanna Di Marzo Serugendo, Sara Montagna, Mirko Viroli, and Josep Lluís Arcos. Description and composition of bio-inspired design patterns: a complete overview. *Natural Computing*, 12(1):43–67, 2013.

[2] Dirk Helbing, Illes J Farkas, Peter Molnar, and Tamás Vicsek. Simulation of pedestrian crowds in normal and evacuation situations. *Pedestrian and evacuation dynamics*, 21:21–58, 2002.

[3] Dirk Helbing and Anders Johansson. Pedestrian, crowd and evacuation dynamics. In Meyers [5], pages 6476–6495.

[4] Ioannis Karamouzas, Peter Heil, Pascal van Beek, and Mark H. Overmars. A predictive collision avoidance model for pedestrian simulation. In *Motion in Games*, volume 5884 of *Lecture Notes in Computer Science*, pages 41–52. Springer, 2009.

[5] Robert A. Meyers, editor. *Encyclopedia of Complexity and Systems Science*. Springer, 2009.

[6] Sara Montagna, Mirko Viroli, Jose Luis Fernandez-Marquez, Giovanna Di Marzo Serugendo, and Franco Zambonelli. Injecting self-organisation into pervasive service ecosystems. *Mobile Networks and Applications*, 18(3):398–412, 2013.

[7] Danilo Pianini, Sara Montagna, and Mirko Viroli. Chemical-oriented simulation of computational systems with Alchemist. *Journal of Simulation*, 2013.

[8] Danilo Pianini with contributions by Michele Bombardi, Chiara Casalboni, Davide Ensini, Sara Montagna, Luca Nenni, Michele Pratiffi, and Mirko Viroli. *The Alchemist simulator - Full manual*.

Ringraziamenti

Un percorso di studi come quello della Laurea Magistrale non è mai qualcosa di banale, tantomeno lo è se lo si segue (o meglio, non lo si segue) in parallelo all'attività lavorativa. Il primo ringraziamento va perciò - doverosamente - al prof. Mirko Viroli e al dott. ing. Danilo Pianini, che mi hanno aiutato nella realizzazione di questo lavoro, soprattutto per aver compreso e rispettato le necessità che la doppia attività lavoro/studio mi poneva. Allo stesso modo ringrazio gli altri docenti e soprattutto il personale della Segreteria Didattica, che si è sempre dimostrato molto comprensivo e disponibile nei miei confronti e che mi ha fatto sentire un po' meno spaesato, nonostante la mia quasi totale assenza - almeno fisica - dagli ambienti universitari. Vorrei in questo frangente ringraziare anche il prof. Roberto Laschi, che fu relatore della mia Tesi di Laurea Triennale, soprattutto per avermi insegnato a dare il giusto peso alle cose - insegnamento non ancora recepito del tutto, ma ci sto lavorando. Grazie a Gaia, per una quantità infinita di motivi ma soprattutto, semplicemente, per esserci. Grazie a Marisa e Giuseppe, che sono la migliore famiglia acquisita che si possa chiedere, per avermi fatto sempre sentire a casa, ai miei genitori e a mio fratello Andrea per il sostegno morale e - quando possibile - per quello materiale, al nonno per la forza che ha sempre avuto nelle avversità, e a tutti i tanti cugini. Non sarei mai potuto arrivare qui senza il necessario sostegno di un nutrito gruppo di amici, grazie quindi ai vari Giovanni, Elisa, Marco, Angelo, Arianna, Giulia, Agnese, Riccardo, Cristina, Federica, Daniela, Federico, a Chiara e Christian, Emanuela e Francesco e a tutti quelli che per motivi di spazio non possono essere qui nominati. Un doveroso ringraziamento va anche ai colleghi di Iprel Progetti, che mi hanno sopportato in questi mesi di doppio lavoro, e agli ex colleghi di Tomware e Santerno. Grazie alle persone che ci hanno creduto, e a quelle che non ci hanno creduto, me stesso in primis: per una volta sono ben felice di ammettere che i primi avevano ragione e gli ultimi, me compreso, torto. Grazie ai Muse, che sono stati la colonna sonora di questi mesi di duro lavoro, in particolare The Resistance e The 2nd Law. Non sarei infine la persona che sono senza le canzoni di Fabrizio De Andrè, i film di Woody Allen, le poesie di Eugenio Montale, i libri di Daniel Pennac, i fumetti di Charles M. Schulz, la musica di Duke Ellington.

Elenco delle figure

Appendice

Codice della simulazione standard

Di seguito è riportato il codice completo della simulazione analizzata nel capitolo 1 e utilizzata per i test i cui risultati sono presentati nel capitolo 4:

Listing 6.1: Simulazione standard

```
1  environment sPietro
2  type PngEnvironment
3  params "15,C:/eclipse/runtime-EclipseApplication/alchemist-
       simulations/src/plants/spietro.png"
4  with linking rule SelectiveEuclideanDistanceWithObstacles params
       "20,0,infrastructure"
5  with random seed 950632084
6
7  lsa source <source, Type, Distance>
8  lsa target <source, target, 0>
9  lsa gradient <grad, Type, Distance>
10 lsa crowd <crowd, L>
11 lsa red <red>
12 lsa green <green>
13 lsa blue <blue>
14 lsa infrastructure <infrastructure>
15 lsa person <person>
16
17 place node at point (144,20)
18 containing red target infrastructure
19 with reactions
20 reaction SAPEREGradient params "ENV,NODE,RANDOM,source,gradient,2,((
       Distance+#D)+(0.5*L)),crowd,2000000,10" []-->[]
21 eco-law compute_crowd []-1-> [agent CrowdSensor params "ENV,NODE"]
22
23 place node at point (144,133)
24 containing red target infrastructure
25 with reactions
26 reaction SAPEREGradient params "ENV,NODE,RANDOM,source,gradient,2,((
       Distance+#D)+(0.5*L)),crowd,2000000,10" []-->[]
27 eco-law compute_crowd []-1-> [agent CrowdSensor params "ENV,NODE"]
28
29 place node at point (37,102)
```

```
30 containing red target infrastructure
31 with reactions
32 reaction SAPEREGradient params "ENV,NODE,RANDOM,source,gradient,2,((
      Distance+#D)+(0.5*L)),crowd,2000000,10" []-->[]
33 eco-law compute_crowd []-1-> [agent CrowdSensor params "ENV,NODE"]
34
35 place node at point (37,77)
36 containing red target infrastructure
37 with reactions
38 reaction SAPEREGradient params "ENV,NODE,RANDOM,source,gradient,2,((
      Distance+#D)+(0.5*L)),crowd,2000000,10" []-->[]
39 eco-law compute_crowd []-1-> [agent CrowdSensor params "ENV,NODE"]
40
41 place node at point (37,53)
42 containing red target infrastructure
43 with reactions
44 reaction SAPEREGradient params "ENV,NODE,RANDOM,source,gradient,2,((
      Distance+#D)+(0.5*L)),crowd,2000000,10" []-->[]
45 eco-law compute_crowd []-1-> [agent CrowdSensor params "ENV,NODE"]
46
47 place 117 nodes in rect (54,40,120,92) interval 10
48 containing in all green infrastructure
49 with reactions
50 reaction SAPEREGradient params "ENV,NODE,RANDOM,source,gradient,2,((
      Distance+#D)+(0.5*L)),crowd,2000000,10" []-->[]
51 eco-law compute_crowd []-1-> [agent CrowdSensor params "ENV,NODE"]
52
53 //moving people
54 place 400 nodes in rect (54,48,120,60)
55 containing in all person blue
56 with reactions
57 []-1->[agent GradientFollowingPredictivePedestrian params "ENV,NODE,
      RANDOM,REACTION,gradient,2,3,20"]
58 //env, node, rand, reaction, target_template, argNumber, loa, range
```