Modelling and simulation of realistic pedestrian behaviours

Koh, Wee Lit

2008

Koh, W. L. (2008). Modelling and simulation of realistic pedestrian behaviours. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/14961>

<https://doi.org/10.32657/10356/14961>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**MODELLING AND SIMULATION OF
REALISTIC PEDESTRIAN BEHAVIOURS**

KOH WEE LIT

School Of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2008

Acknowledgements

I would first like to express my thanks for Dr. Zhou Suiping, my supervisor, for his guidance and foresight during the my last two years with him. In particular, he has brought me to see the different perspectives of things that are always possible for any given situation. Without his patience and sincere support, my research will definitely have been much more difficult.

Next I will like to thank Dr. Wong Kok Cheong, my one time supervisor, who have actively encouraged me to take up graduate studies as a Masters student. He has also been very supportive of my later conversion to a PhD candidature.

I am also grateful to the staff and lab technicians in Centre of Graphics and Imaging Technology, Temasek Laboratories @ NTU and the Parallel & Distributed Center for their support. Special mention is also appropriate for Alan Chng, Chua Beng Hwee and Lin Lin for the roles they have played.

Finally, I am grateful to my parents and brother, for their support, both morally and financially, throughout my candidature. Without them, this thesis would never have become a reality.

Koh Wee Lit
June 2008

Contents

1	Introduction	10
1.1	Background	10
1.2	Objectives of Study	16
1.3	Organisation of Thesis	17
2	Methodologies for Representing Pedestrian Crowd	19
2.1	Flow Based Approaches	19
2.2	Celluar Automata Approaches	23
2.3	Particle Based Approaches	28
2.4	Agent Based Approaches	31
2.5	Distributed Approaches	36
2.6	Summary	38
3	Representation of A Pedestrian	41
3.1	Methods and Considerations of Agent Representation	43
3.1.1	Body Representation	44
3.1.2	Sensory Representation	46
3.1.3	Environmental Awareness Representation	48
3.1.4	Inter-agent Awareness Representation	51
3.1.5	Sensory and Visual Memory Representation	52
3.1.6	Attention Representation	54
3.2	Agent Representation of a Pedestrian	56
3.2.1	Body Representation	57
3.2.2	Sensory Representation	60
3.2.3	Sensory and Visual Memory Representation	65
3.2.4	Attention Representation	68
3.3	Summary	69
4	Micro-Level Navigation Behaviour for Pedestrians	72
4.1	Micro-Level Navigation Model	75
4.1.1	Collision Prediction	78
4.1.2	Collision Avoidance Constraint	81
4.1.3	Speed Constraint	88
4.1.4	Orientation Constraint	90
4.1.5	Combining Constraints	90
4.1.6	Resolving Conflicting Constraints	94
4.2	Summary	97

5	Macro Level Navigation Behaviour for Pedestrians	100
5.1	Methods and Considerations for Path-Finding	102
5.1.1	Representation of Environment for Path-Finding	103
5.1.2	Methods for Path-Finding	110
5.1.3	Considerations for Pedestrian Simulation	116
5.2	Macro-Level Navigation Model	121
5.2.1	Representation of the Environment	122
5.2.2	Heuristics for the Path-finder	128
5.3	Path-Finding and Visual Memory	130
5.4	Summary	133
6	Modelling and Simulation of Dense Pedestrian Crowd	135
6.1	Extending the Micro-Level Navigation Module	137
6.1.1	Limitations of the Micro-Level Navigation Module . . .	137
6.1.2	Extended Collision Avoidance Constraint	140
6.2	Pedestrian Behaviours in a Dense Crowd	143
6.2.1	Lane Formation Behaviour	143
6.2.2	Queueing Behaviour	146
6.2.3	Pushing Behaviour	147
6.2.4	Falling and Trampling Behaviour	148
6.3	Summary	149
7	Implementation and Experimental Results	151
7.1	Memory Management	152
7.1.1	Allocation Strategies	154
7.1.2	Memory Wastage	156
7.1.3	Search Performance	157
7.1.4	Memory Management for Crowd Simulation	158
7.2	Simulation Scenarios	161
7.2.1	Wide Open Spaces	161
7.2.2	Narrow Spaces	163
7.2.3	Indoor Areas	164
7.2.4	Urban Areas	166
7.3	Basic Pedestrian Behaviours	168
7.3.1	Simple Avoidance Behaviour	168
7.3.2	Seperation Behaviour	170
7.3.3	Speed Matching Behaviour	172
7.3.4	Overtaking Behaviour	173
7.4	More Complex Behaviours	174
7.4.1	Two Groups	174
7.4.2	Three Groups	175

7.4.3	Social Status	176
7.5	Dense Pedestrian Crowd Simulation	177
7.5.1	Ad-Hoc Lane Formation	178
7.5.2	Voluntary Lane Formation	179
7.5.3	Effects of Personal Space	180
7.6	Pedestrian Simulation in Complex Environments	182
7.6.1	Path-finding Capability	182
7.6.2	Congestion Avoidance	184
7.6.3	Shopping Mall	185
7.7	Real-Time Performance Analysis	187
7.7.1	One-Directional Pedestrian Flow	189
7.7.2	Four-Directional Pedestrian Flow	190
7.8	Performance Optimisation Using Free-lists	192
7.8.1	Free-Lists for Path-Planning	194
7.8.2	One-Directional Pedestrian Flow	195
7.8.3	Four-Directional Pedestrian Flow	196
7.9	Performance Optimisation Using Time Slicing	199
7.9.1	Using Buckets to Improve Performance	200
7.9.2	Trade-Offs For Using Time-slicing	204
7.9.3	Time-Slicing for Different Levels of Detail	207
7.10	Summary	209
8	Conclusion	211
8.1	Contributions	212
8.2	Future Challenges	214
	References	217
	Appendix A Proof of the Collision Speed Constraint	230
	Appendix B Proof of the Multi-agent Avoidance Constraint	235
	Appendix C Author's Publications	241

List of Figures

3.1	Agent representation.	60
3.2	Agent vision model.	62
3.3	Vision range and sample density.	63
3.4	A relative frame.	67
3.5	Representation of an autonomous agent.	70
4.1	Proposed representation of an autonomous pedestrian.	73
4.2	Collision zone between the source and target agent.	79
4.3	Collision prediction.	82
4.4	Possible new velocities of the source agent.	84
4.5	The Duality Property.	85
4.6	Use of an orientation constraint.	91
4.7	Normalising the constraints.	92
4.8	Combining different constraints.	92
4.9	Pseudocode for computing the constraint intersection.	93
4.10	Conflicting constraints.	95
4.11	Prioritising constraints.	96
4.12	Micro-level navigation model.	98
5.1	A corner graph for path-finding.	107
5.2	A circle graph for path-finding.	108
5.3	Congestion avoidance.	113
5.4	Macro level navigation model.	122
5.5	Construction of a Quadtree.	124
5.6	Basic connectivity.	125
5.7	Trace of route generated.	125
5.8	Proposed waypoint layout.	126
5.9	Sample connection between regions.	127
5.10	Navigation around corners.	127
5.11	Visual memory and route choices.	132
6.1	Peak hours crowd at Penn Station, New York.	136
6.2	Voluntary lane formation in Shinjuku Train Station, Tokyo.	138
6.3	Three relative frames between different target agents.	140
6.4	Lane formation behaviour.	145
6.5	Ad-hoc lane formation during the Tokyo Game Show.	146
6.6	Ticket queuing at Suzhou Railway Station, China	147
6.7	Pedestrians pushing to board a train.	148
7.1	Pseudocode for memory management using <i>free-list</i>	159
7.2	Pseudocode for the expanding the <i>free-list</i>	159
7.3	Tian'an men Square in Beijing, China.	162
7.4	Pedestrian simulation in wide open spaces.	162

7.5	Crowded street along a popular shopping district.	163
7.6	Pedestrian simulation in simple enclosed spaces.	163
7.7	Interior of a typical shopping mall.	164
7.8	Pedestrian simulation in a shopping mall.	165
7.9	A pedestrian zone in central Munich, Germany.	167
7.10	Pedestrian simulation in a urban area.	167
7.11	AABB, personal space and vision samplers.	169
7.12	Collision avoidance behaviour.	170
7.13	Separation behaviour.	171
7.14	Another way of avoiding the same collision.	171
7.15	Speed Matching behaviour.	172
7.16	Overtaking behaviour.	173
7.17	Two-way pedestrian crossing.	174
7.18	Three-way pedestrian crossing.	175
7.19	Different social status.	176
7.20	Simulated Ad-hoc lanes.	178
7.21	Voluntary lane formation at Batu Caves, Malaysia.	179
7.22	Simulated voluntary lane formation.	179
7.23	Crowd with different density levels.	181
7.24	Effects of the agent's personal space.	181
7.25	One directional pedestrian flow.	183
7.26	Four directional pedestrian flow.	183
7.27	Use of detours to avoid congestions.	184
7.28	Congestion avoidance in a shopping mall.	185
7.29	Agents with different roles in a shopping mall.	186
7.30	Agent behaviour in one directional pedestrian flow.	190
7.31	Render period for one directional pedestrian flow.	191
7.32	Agent behaviour in four directional pedestrian flows.	192
7.33	Render period for four directional pedestrian flows.	193
7.34	Agent behaviour using freelists.	196
7.35	Render period using freelists.	197
7.36	Agent behaviour using free-lists.	198
7.37	Render period using free-lists.	198
7.38	Agent behaviour using two buckets.	200
7.39	Render period using two buckets.	201
7.40	Agent behaviour using two four buckets.	202
7.41	Render period using four buckets.	202
7.42	Agent behaviour using eight buckets.	203
7.43	Render period using eight buckets.	203
7.44	Render period using different number of buckets.	205
7.45	Improving performance using Time-slicing.	207

7.46 Time-slicing and Level of Detail. 208
A.1 Speed constraints in terms of ω_s 233
B.1 Combined vector diagram of the two relative frames. 236

Summary

Pedestrian simulation provides interesting challenges in the area of 3D visualization, computer animations, behavioural modelling and real-time crowd simulation. With increasing computing power, many researchers are now proposing autonomous agent representation and behavioural models that more closely reflect the behaviours of real-humans. Following this trend, an accurate representation of an autonomous agent that has human-like sensory inputs and attention systems has been developed in this research. Another important consideration in this research is to balance between the accuracy of a pedestrian representation and the computational costs of using such a representation.

A pedestrian simulation is primarily concerned with the generation of realistic navigation behaviours for autonomous agents. To address this challenge, a two-tier navigation model for path-planning and reactive planning in a complex and dynamic environment has been proposed to more accurately model the navigation behaviours of a pedestrian. First, a macro-level navigation model is developed to mimic the way pedestrians plan their route to reach a particular destination in the environment. Secondly, a micro-level navigation model is developed to steer the agent away from collision with structural obstacles and other agents. In addition, the micro-level navigation model ensures that the agent stays on course according to the rough path computed by the macro-level navigation model. The proposed navigation model is able to combine the different navigation considerations of a pedestrian in the form of steering constraints on the micro-level navigation

model and heuristics for the macro-level navigation model. Moreover, the navigation model is computationally efficient in that it is able to compute hundreds of agent movements in a fraction of a second.

The proposed pedestrian representation and navigation model reflect the various factors in human navigation behaviours such as patience and age. As illustrated by the simulation studies, the proposed models are able to generate various pedestrian behaviours that can be seen in daily life. In particular, the lane formation behaviour of pedestrians in a dense crowd can be observed naturally using the proposed navigation model. In order to improve the performance of the pedestrian simulation, two performance optimising techniques has been proposed. First, free-list has been used to reduce the computational cost of allocating new memory to the simulation. Secondly, time-slicing has been used to reduce the duration of each update cycle by grouping individual agents into buckets. The performance of the proposed optimisation techniques has also been evaluated.

1 Introduction

1.1 Background

The modelling and simulation of realistic pedestrian navigation behaviours aims to describe and predict pedestrian movements through different environments such as urban areas and inside buildings, and in different situations, such as peak hours and crisis evacuations. Realistic pedestrian navigation behaviours for autonomous agents can be achieved by first observing the movements of real pedestrians. Based on these observations, it will then be possible to experiment, explain and generate these observed behaviours using a navigation model. In particular, we are interested in developing a navigation model for autonomous agents that is capable of describing realistic pedestrian navigation behaviours such as collision avoidance, overtaking, lane following and congestion avoidance.

In real-life, pedestrians will influence how other pedestrians will react with their own walking behaviours. They need to adopt a variety of maneuvers such as speed matching and overtaking to avoid any potential collision with other agents, to reach their own destination. At the same time, different pedestrians have different personalities which will in turn influence their navigation choices. For example, some pedestrians may prefer to maintain their speed throughout their navigation while other pedestrians may be reluctant to step aside to avoid others. Pedestrians also have their own preferences such as a comfortable walking speed, and their own habits, such as the continuous use of the same route to reach the same destination. They also have their own perception of the environment, i.e., usually incomplete knowledge

about the environment and sometimes ignorant of the shortest possible path to a destination.

Equally important are that pedestrians may have their own subjective judgement. For example, what is considered fast to a pedestrian may be considered slow to another pedestrian. Pedestrians will also behave differently in different social context and around different social groups. For example a child will only follow their parents and not any other pedestrian nearby. People have different expectations of how other people around them will move in response to own movements. For example, important people like a country's prime minister will expect to have the *right-of-way*, i.e., other pedestrians are expected to make way to the prime minister rather than the other way around. Different cultures will also have different social norms that will influence how a pedestrian will move, e.g., keeping to the left (or right) side of the lane when moving. For a realistic pedestrian simulation, it will therefore be important to understand and incorporate these differences among individual pedestrians and understand how these factors will influence the navigation choices of different people. However, the main challenge will be the integration of these factors in an effective manner to generate realistic pedestrian behaviours, i.e., how to combine considerations from different domains such as patience and comfortable walking speed together to describe the mental state of a pedestrian.

Pedestrian simulation can be used to evaluate the safety [36], comfort [101], [129], and efficiency [44], [83], of different structures and environments. For example, pedestrian simulation can be used to simulate egress situations such as [100] and [99] to observe and study how pedestrians move under nor-

mal and panic situations. Building designs can hence be first modelled as a virtual environment and tested with computer generated agents. The designs can then be evaluated cheaply and easily before the actual commencement of the construction.

Pedestrian simulation can also be used to automatically generate a pedestrian crowd [94], [111], [120], with realistic navigation behaviours in animation systems or in interactive digital medias such as massively online multi-player games. Realistic pedestrian simulation is important in this case to generate a pedestrian crowd that look natural and realistic. Using an automated crowd animation system can also help to reduce the production time and cost required to create a visually pleasing pedestrian crowd.

For a pedestrian simulation, it is equally important for the generated pedestrian behaviours to be realistic as it is computationally efficient to generate them. For example, it will be useful to allow the size and composition of the pedestrian crowd to be changed dynamically and then to receive immediate feedbacks on the results of such changes on the adequacy of evacuation routes from a building. Similarly, for applications in computer games, it will be important to ensure that the game can continue to run with a reasonable frame rate when using our model to generate a realistic pedestrian crowd.

May [75], proposed that pedestrian simulations can be divided into macroscopic and microscopic levels. Microscopic level simulations aim to observe the movements of individual agents within the simulation while macroscopic level simulations look at the high level flow of pedestrians without identifying individual agents. In this thesis, we are primarily interested in the study of microscopic pedestrian simulations and the behaviours of in-

dividual pedestrians with respect to the movement around them. At the same time, no discussions about microscopic pedestrian simulation can be complete without the modelling of a pedestrian crowd. This is because some pedestrian behaviours such as lane formation, can only be observed in a crowd setting.

Therefore, the modelling and simulation of realistic pedestrian navigation behaviours is an interesting challenge because the simulation needs to focus on the navigation behaviours of individual pedestrian and at the same time require many technologies used in a crowd simulation to generate a real-time pedestrian crowd. On one hand, we need to develop approaches to describe the movement of autonomous agents to mimic that of a pedestrian. For example, how a pedestrian moves in a busy intersection will be different with how a pedestrian moves in a shopping mall. On the other hand, we need to ensure that the agent's movement is computationally efficient enough to allow the parameters within the simulation to be changed dynamically. In particular, the agent's navigation model must be able to compute the movements of hundreds of agents in a fraction of a second

One of the pioneering works that focus on the modelling realistic behaviours for autonomous agents originates from Thalmann [115]. Thalmann argued that virtual human agents being modeled after real humans should be able to act and react to a dynamic and unpredictable virtual world. Autonomous agents should therefore be as realistic and spontaneous as possible within the constraints of a real-time simulation.

More recently, Teknomo pointed out that the way real pedestrians move around in the environment depends not only upon their own intention but

also the behaviour of other pedestrians around them [112]. For example, pedestrians are expected to change their current headings in order to avoid or overtake other agents. Teknomo's uses a Newtonian approach to model the social attractive and repulsive forces experienced by the agent to control the steering parameters of the agent. Teknomo's approach considers navigation as a result of *social forces*, as first proposed by Helbing [44]. That is movement of a pedestrian is a result of the desire to move away from other pedestrians (repulsive social force) and the desire to move towards a certain goal (attractive social force). However, it is important to note that the derivation for the magnitude and direction of these social forces have not been given sufficient consideration. At the same time, social forces and physical forces are from different domains. As a result, it may be difficult to combine these forces together to generate meaningful behaviours.

Sakuma proposed an agent model using human psychological models. In particular, memory and vision has been considered in Sakuma model [95]. Pedestrian navigation is then achieved using *potential fields*, as first proposed by Reynolds [92]. For the decision to avoid collision, Sakuma proposed the use of a two stage personal space. When agents are found in the inner personal space, these agents should be avoided immediately. On the other hand, agents found in the outer personal space represents agents that should be avoided without taking sudden maneuvers. The inclusion of individual memory and vision will help ensure that the generated behaviours will be varied among different agents. This is an important property which we will address throughout our study. This is because we believe that the generation of different reactions by different individuals in a similar scenario is an im-

portant property to create a believable pedestrian crowd. However Sakuma's agent model lacks a macro-level planning layer and may have difficulty in handling complex environments. That is the model does not consider the macro-level behaviours of a pedestrian, such as seeking alternative routes to the destination in the case of congestions during navigation, which may influence the route choice of a pedestrian.

Therefore in this thesis, we will focus our discussions on the modelling and simulation of basic navigation behaviours that can be combined to generate a pedestrian's navigation behaviour for different environments and scenarios. We then aim to use these basic behaviours to realise more complex social behaviours like lane formation and group movements. We will also propose and implement some optimisation techniques to reduce the time necessary to compute the movements for the autonomous agents. Our main contribution towards the modelling and simulation of pedestrian behaviours can best be summarized a bridge between physio-sociological studies of real pedestrian behaviours such as work of Baddeley [5], Gwynne [36], Posner [87], Rao [91] and Rymill [93], with the models for generating automated agent movement and animation such as work of Burstedde [16], Helbing [41], Reynolds [92], Schadschneider [97], and Thalmann [115]. Achieving realistic navigation behaviours for our autonomous agents is the goal of our research.

1.2 Objectives of Study

Having understood the applications and challenges of a pedestrian simulation, we can now better define the objectives of our study in this thesis. Specifically, our objectives are:

1. To create a microscopic pedestrian simulation capable of generating realistic pedestrian navigation behaviours;
2. To provide new insights on how and why pedestrians move in a particular manner;
3. To generate various pedestrian phenomena like *lane-formation* by allowing pedestrians to *self-organise*;
4. To propose an accurate representation of real pedestrians for use in a real-time pedestrian crowd simulation;
5. To propose a new navigation module capable of describing realistic navigation behaviours for self-driven autonomous agents that considers the mental state of real pedestrians;
6. To study and propose optimisation techniques that will improve the performance of a pedestrian crowd simulation.
7. To evaluate the performance improvements and potential trade-offs for using the proposed optimisation techniques.

1.3 Organisation of Thesis

We have discussed our motivations, objectives and requirements for the modelling and simulation of a real-time pedestrian crowd. The rest of this thesis is organised as follows:

1. Chapter 2 will review existing methods of representing and simulating a pedestrian crowd;
2. Chapter 3 will examine how an accurate representation of an autonomous agent can be constructed;
3. Chapter 4 will discuss the navigation modules required by our agents to demonstrate realistic navigation behaviours. In particular, we will discuss a novel micro-level navigation module capable of allowing our agents to dynamically react to the movement of other agents and changes in the environment;
4. Chapter 5 will discuss the macro-level navigation module that is able to help our agents navigate through different complex environments;
5. Chapter 6 will examine the simulation of dense pedestrian crowds. In particular, we will examine pedestrian behaviours typical in a dense pedestrian crowd, like lane formation, queuing and pushing;
6. Chapter 7 will report the results of our experiments and demonstrate the various navigation behaviours that emerge as a result of using our proposed two-tier navigation module. We have also implemented two performance optimisation techniques, a free-list memory manager and

the use of time-slicing. Finally, the runtime performance of the simulation will be analysed.

7. Chapter 8 will conclude our study and propose potential future directions of our work.

2 Methodologies for Representing Pedestrian Crowd

One of our objectives in the modelling and simulation of pedestrian behaviours, is to be able to reproduce different interesting pedestrian behaviours and pedestrian flows. In this section, we will review several existing methods that can be used to model realistic pedestrian behaviours with different level of details. In particular, we are interested in pedestrian models that are capable of representing the physiological states of a real pedestrian and generating realistic navigation behaviours for individual agents.

2.1 Flow Based Approaches

Flow based approaches are macroscopic models that do not concern with the movement of individuals within the crowd [47], [49]. Instead, they model the pedestrian *flow* directly, i.e., by modelling the environment as a large *flow network* and using flow equations to describe the rate at which pedestrians can move through different regions of the environment. The flow network consists of a series of corridors and junctions to represent regions in the virtual world that are accessible to the agents. At these junctions, nodes are then added and connected to create a graph-like structure. There are two main types of nodes in a flow network, source nodes and sink nodes. *Source nodes* are nodes that introduce flows into the network while *sink nodes* are nodes that allow flows to exit the network. The flow direction of agents in a flow network may be unidirectional, bi-directional or unconstrained.

Rules derived from both empirical studies and real-life observations can then be used to regulate both the current and maximum flow rate between the source and sink nodes. Obstructions to the pedestrian flow can also be introduced to study its effect on the entire flow network. For example, during a crisis scenario, a *fire* in one of the corridors may be represented as a severing of the a connection between two nodes. As a result, traffic from the neighbouring links will need to divert and find alternative routes. In this case, the flow network can be tested for its capacity to handle the strain on its flow capacity as a measurement of design efficiency. The advantages of using a flow based approach to modelling a pedestrian crowd

1. Simplifying the modelling of the environment and allowing the study to focus on different characteristics of the pedestrian flow;
2. Modelling the flow of an entire city without the use of sophisticated hardware;
3. Integrating the results from empirical studies to support and customise a flow based model;
4. Fitting a mathematical equation to describe the pedestrian flows, and may hence provide new insights about how pedestrians move.

Due to the high level view provided by this simulation methodology, flow based approaches may be useful in

1. Evaluating the adequacy of a public transport system by studying the pedestrian and traffic flows through a city;

2. Studying pedestrian flows through different areas of a shopping mall to evaluate the layouts of shop in the mall;
3. Analysing the adequacy of evacuate routes such as stairs for a high-rise office building.

Flow based simulations frequently use physical laws such as the *Gas-Kinetic Approach*, as described by Prigogine to model the flow rate along the different branches of the network [90]. Although the approach has been proposed to model vehicular traffic, it is still one of the pioneering works on flow based crowd simulation. The model considers the density and velocity of the traffic as a whole and subjects them to various flow equations. The model does not take into consideration low level vehicle maneuvers such as lane-changing. By considering empirical data collected from real traffic conditions, differential equations are then used to describe the changes in the density of the road traffic. For example, a reduction in the traffic density implies that the traffic can now travel faster. Similarly, an increase in traffic density will imply that a congestion has occurred.

The shortcomings of Prigogine’s approach were pointed out by Hoogendoorn, who also described a new model that targets at *mesoscopic* simulations [48]. i.e., a simulation with level of details between that of a macroscopic and microscopic simulation. Hoogendoorn’s approach is not a microscopic model because the approach still does not distinguish or trace individual entities within the crowd. Instead, the model specify the behaviours of individuals using only probabilistic terms. For example, Hoogendoorn models an agents decision to change its speed as a probabilistic factor that affects the overall

flow rate of the agents through a location in the environment.

As pointed out by Hughes, pedestrians are way too complex to be represented as a singular entity [51]. For example Hughes observed that humans being very goal-oriented, will attempt to avoid congestions by taking de-tours instead of following the shortest path at all times. Therefore, Hughes argued that there should be more than one type of agents in the crowd, i.e., a crowd composed merely of clones from the same basic agent model is not sufficient in the modelling of a realistic pedestrian crowd. However in Hughes' model, the composition of several types of pedestrians into the flow equations is not trivial. Complex and computationally expensive mathematical equations are required to simulate a pedestrian flow that is composed of several different types of pedestrians.

In some situations, the level of details provided by a *macroscopic model* such as the flow based approached may not be sufficient. This is especially true if the movements of individual agents in the crowd needs to be traced by the application. As a result, flow based approaches are less suitable for microscopic pedestrian simulations because flow based approaches are not able to

1. Model the microscopic inter-pedestrian interactions during pedestrian navigation.
2. Model the fact that individuals may choose to move in the environment differently from other agents due to their distinct personalities.
3. Produce realistic trajectories for individual agents for use in automated animation systems.

4. Model the high-level cognitive capabilities, such as path-planning, of individual autonomous agents.

In particular, individual mental states and social needs cannot be addressed effectively in a flow based model. For example, flow based models may have some difficulty integrating the possibility that real humans may not necessarily want to reach the evacuation point at the first instance of a crisis. Some people may want to retrieve their valuables while others might want to reach for their dear ones first before the evacuation.

2.2 Cellular Automata Approaches

Cellular Automata is a widely used approach capable of representing individual agents within a crowd simulation. Cellular Automata works by first dividing a given environment into a finite number of discrete cells. Each of these cells is then given a set of simple rule that controls the information that is contained within these cells. These states may contain simple *on* or *off* information to represent whether an agent is within the cell, or they may contain more complex information such as the number of agent that have recently used the cell. The advantages of using cellular automata are that they

1. Divide the environment into regular grids allowing the environment to be represented easily and cheaply using an array.
2. Allow rapid setup of the environment without having the need to create elaborate environments.

3. Can allow multiply layers of the environment, each representing different pieces of information to be combined together for analysis.
4. Reduce the movement degree of freedom for individual agents, hence allowing for simple rules and models to create realistic pedestrian flows on sufficiently fine grids.
5. Naturally allow for emergent behaviours to surface in the simulation.

Cellular automata's unique positioning between macroscopic and microscopic pedestrian simulation allows it to be used for

1. Modelling of various pedestrian phenomena such as queuing and jamming for a dense crowd simulation [16], [26];
2. Simulation of pedestrian evacuation behaviours such as herding [57];
3. Simulation of kinship behaviours among autonomous pedestrians during normal and crisis simulations [126].

The origins of cellular automata can be traced to the work of Conway's Game of Life [29]. In the case of Conway's game, the rules govern whether each cell lives or dies depending on the state of its neighbors. Each cell in Conway's game relies on four simple rules:

1. Any live cell with fewer than two live neighbours dies due to loneliness.
2. Any live cell with more than three live neighbours dies due to overcrowding.
3. Any live cell with two or three live neighbours lives, unchanged, to the next generation.

4. Any dead cell with exactly three live neighbours comes to life.

Using these rules, the game was able to generate new and interesting patterns although these patterns were never part of the rules. Cellular automata has therefore demonstrated that *emergence behaviour* and *self-organization* can be achieved by applying a simple set of rules on each individual.

However the real link between cellular automata and autonomous agents only came later from Wolfram [125]. He noted that the behaviours demonstrated by cellular automata falls into four classes:

1. Evolution leads to a homogenous state where all cells in the simulation are of the same state. as if by loneliness.
2. Evolution leads to a set of simple stable or periodic structures like the patterns exhibited by Conway's Game of Life.
3. Evolution leads to chaotic pattern where the pattern does not repeat itself with sufficient frequency.
4. Evolution leads to complex localized structures, sometimes long-lived.

The fourth behaviour demonstrated by cellular automata systems is of most interest to our study. This behaviour proves that by focusing on the development of simple behaviours for each agent in the crowd similar to the way rules are applied to autonomous cells in cellular automata systems, it may be possible to demonstrate different localize pedestrian patterns in a crowd. These patterns may for example include land formation and group movements. Furthermore these patterns will not always repeat itself and is therefore able to generate a more natural looking pedestrian crowd.

Blue modelled pedestrian behaviours using cellular automata as an alternative to flow based simulations [10]. Blue pointed out the difficulty of using and tracking flow based simulations for pedestrian simulation. A simpler and more visually pleasing approach of simulating pedestrian flows was therefore proposed using cellular automata. Blue proposed two rule sets to model the movements of real pedestrians. The first set of rules address the lane changing behaviours among pedestrians as a means to get to the destination faster. The second set of rules defines how a cell may be turned on or off to represent the movement of pedestrians through the cell. Each pedestrian will look for a maximum of eight cells forward to determine if it should move forward or *side-step* to change lane. Blue's approach allowed for ad-hoc lane formation because each pedestrian is given the alternative to change lane and therefore form a new lane. There were no constraints to force pedestrians to for example move only on the left or right side of a street. However, Blue's probabilistic lane changing behaviour is not realistic for a microscopic pedestrian simulation. For example, an agent's decision to change lane occurs when the agent in-front is moving too slowly relative to the agent, i.e., the decision to overtake depends upon the speed constraints of the overtaking agent. Therefore, Blue's model does not realistically represent the true mental states of a pedestrian.

Schadschneider proposed another interesting use of cellular automata for agent simulation [97]. In Schadschneider's model, each agent will leave a *trail* of information for the other agents in the scene, much like the chemotaxis phenomena found in ants. Other agents can then decide to follow the trail or to leave it alone. If the other agents decide to follow this *trail* then its

scent will be reinforced. At the same, time, if other agents decide to ignore the trail, then its scent will decay over time and eventually disappear. The proposed model is therefore able to show lane formation phenomena when there are two or more directional flows in the environment. The main issue about Schadschneider's model is the reinforcement and decay rate of the *trail*. If the reinforcement rate of the trail is slow, then lanes will also be formed slowly. Similarly, if the decay rate is too fast, then the trail will dissipate the moment they are formed. Therefore Schadschneider's approach may not be suitable for modelling pedestrian flows in a rapidly changing environment.

Due to the fact that cellular automata models always discretise the environment into regular grids, current cellular automata models may not be suitable for modelling a microscopic pedestrian simulation because it is

1. Unable to trace the movement of individual pedestrians with details necessary to create a visually pleasing close up view of the pedestrian simulation;
2. Inheritably rule-based and therefore does not naturally provide new ways of describing the movement of a pedestrian;
3. Difficult to visualise exactly how a movement is executed due to the discrete moving of individual pedestrians.
4. More interested in describing the various pedestrian phenomena rather than representing the actual mental states of a pedestrian.
5. Memory inefficient for large and complex environments due to the grid-like division of the environment.

2.3 Particle Based Approaches

Particle based approaches aim to model the crowd as particles that obey Newton's Law of Motion. However, unlike the flow based models, each agent in a particle based approach is now modelled as a separate particle with its own velocity and acceleration. The particles are then moved in a force field similar to fluid dynamics. These forces are sometimes termed as *Social Forces* to represent the fact that the pulling and pushing forces mimic the actual social interactions between the agents. The advantages of using a particle based approach are that the approach allows

1. Modelling and simulation of individual agents with full degrees of movement freedom on a planar floor.
2. Use of physical laws to change the behaviours of individual agents allowing for smooth trajectories.
3. Use of existing physical models to combine and solve different physical constraints.
4. Realistic modelling of pedestrian navigation based on internal (self-driven) and external (social forces) factors.

As a result, particle based approaches are well suited in the modelling of

1. Social interaction between different pedestrians simplified as pull and push social forces.
2. Collision avoidance between different pedestrians using repulsive social forces.

3. Goal oriented movements for individual pedestrians using attractive social forces.
4. Pushing and trampling of pedestrians during evacuation scenarios and during peak-hours.
5. Automated pedestrian animation systems for simple unidirectional pedestrian flows.

Helbing's work on traffic simulation is one of important contributions to the area of particle based simulation [44]. Specifically, the concept of self-driven particles have made Helbing's work distinct from conventional physical systems. This is because, unlike physical interactions between inanimate objects, Helbing's particle approach for example models that self-driven particles will put up an opposing force of different intensity based on personal strength. Helbing studied on two key areas of particle flows, vehicular [41] and pedestrian [42], [43]. Helbing's work on pedestrian simulation aims to identify potential flow bottlenecks in building layouts and pedestrian facilities that may prove to be hazardous for some pedestrians in a dense crowd. Helbing argued that the social force model is capable of reproducing various emergent pedestrian phenomena in pedestrian crowds and that these behaviours are important for the optimization of pedestrian flows.

Several existing works [67], [128], have also either used or extended parts of Helbing's particle model. For example, Kirchner used a particle approach to model the queueing behaviours of pedestrians in a dense crowd simulation by introducing a *friction* factor [58]. *Friction* in this case refers to the probability that an agent will be unable to move in the current turn as a re-

sult of inter-agent pressure when modelling a dense pedestrian crowd. Gloor simplified Helbing’s particle model to describe the movements of individual agents in an open terrain [30]. In particular, Gloor’s model used particles for microscopic or closed up view of the simulation in example applications like hiking in the Swiss Alps.

One interesting note about Helbing’s social force approach is that an *external force* is used to model a pedestrian’s navigation behaviours, although such an external force does not physically exist. As a result, Helbing’s model does not realistically represent why a pedestrian move in a particular manner. By comparison, our proposed approach models the agent’s navigation behaviour as a series of physiological forces that originates from the the agent itself, which in turn is then affected by various external factors such as the position and orientation of other agents. In addition, Helbing’s model may not be suitable for microscopic pedestrian simulation because it

1. Relies on virtual social forces to propagate individual through the crowd rather than the actions of individual agents;
2. Uses second order differential equations such as the agent’s acceleration, to describe and control the movement of pedestrians and is hence less intuitive for users than if the agent’s speed and orientation are used;
3. May result in deadlocks between agents as a consequence of conflicting social forces;
4. May require several trail and error attempts to obtain the correct amount of social force acting on an agent for different situations and environments.

2.4 Agent Based Approaches

Agent based approaches models a pedestrian as an autonomous agent. When placed together in a computer simulation targeted at replicating a particular scenario such as pedestrian evacuation, an agent based simulation can be used for simulating the actions and behaviours of the agents within the scenario. Hence, the motivations for applying an agent based model to represent individuals within a crowd are its ability to

1. Focus on the various behavioural aspects of the entity that are important in a particular scenario;
2. Model what an individual agent is capable of perceiving in the virtual environment;
3. Model micro-level behaviours such as steering and macro-level behaviours such as planning for individual agents;
4. Naturally allow emergent behaviours to emerge at a crowd level as a result of applying only basic rules to individual agents;
5. Model individual decision making process as a result of previous experiences and personal choices.

Due to the unique approach of modelling individual agents as a self-contained entity, agent based simulations can be used to

1. Create automated animation systems to synthesize the motions of individual agents affected by physical constraints;

2. Evaluate the efficiency of various building layouts and pedestrian facilities;
3. Evaluate the safety and adequacy of evacuation routes from a specific office floor or high density residential block;
4. Gain new insights on how pedestrian move in different situations, environments and social context;
5. Model interesting tactical behaviours for computer generate forces in a military simulation.

There are several existing approaches for modeling real-life entities as individual software agents. For example, the flocking approach is a form of agent based simulation where agents are given a set of simple rules, sufficient to avoid collision and move in the general direction of the group. Depending on the real-life entity being represented by each agent, the rules can have different composition and complexity. However unlike cellular automata, a flocking approach allows for seamless transition between different levels of detail in real-time. For example, the simulation can be zoomed in to view various agent-level interaction or zoomed out to view the entire pedestrian flow.

One of the earliest works of flocking simulation can be traced to Breder [13]. Breder through the analysis of empirical biological data fitted a *potential field* around individual fishes. The concept of *attractive and repulsive potential* was then formulated as a *pseudo-force* that keeps the school of fish from colliding with each other and at the same time maintains the school's

topology. This approach is similar to Helbing’s particle model except that a heuristic or approximate function is used to drive the movements individual agents in flocking simulation rather than functions describing the movement of fluids.

The notion of *potential-based motion control* gained popularity when Reynolds published his highly influential paper titled "Flocks, Herds and Schools: A Distributed Behavioral Model" [92]. By treating each individual agent within the flock as a particle coupled with a potential field model, Reynolds allowed the simulation of realistic flocks that demonstrates behaviours such as leader following and flock aggregation. The potential field approach basically attaches a bounding volume around each software agent. Agents navigating near the bounding volume will self-generate a repulsive force of pre-determined magnitude to push themselves away from the bounding volume. This force is usually known as the *repulsive potential* of the obstacle. Like a flock of birds, agents are also expected to have a spatial goal for which they will want to reach. A similar but opposite force is hence introduced to attract the agents towards the goal. This force is also known as the *attractive potential* of the goal. A typical agent will then rely on these push and pull potentials to navigate through the environment, thereby avoiding both structural obstacles and other agents.

Although Reynold’s work is meant to replicate the behaviours of flocks of birds and herds of animals, several recent works [20], [38], have successfully applied the *potential field* approach to model the movement of real humans. For example, Treuille introduced the concept of dynamic potential fields that consists of both macro-level path-planning and micro-level collision avoidance

into one framework [119]. One important difference between Treuille’s work and Helbing’s particle model is that, Treuille’s model considered the personal preferences of individual agents when selecting a path they want to use when moving through the environment. This preference is modelled as a *discomfort field* that pushes the agents away from a path that they are uncomfortable or unfamiliar with. Added to the fact that it is computationally cheap to compute the steering parameters of an agent using a flocking model, Treuille’s model is able to support tens of thousand agents on a single workstation with a reasonable frame rate. Such a model will therefore be suitable for modelling a large *homogeneous* crowd.

However there are several setbacks for using a potential field approach as raised by Koren [64]. For example, a potential field approach suffers from the inability to resolve conflicting forces effectively. In this case, the movement of agents may become oscillatory and unstable. Furthermore, potential field models while trivial to implement, lacks intuitive controls for users to generate a specific navigation behaviour. As a result, this may have led to a wide variety of agent based alternatives being proposed to describe the movement of pedestrians.

For example, Loscos proposed an easy to understand rule based approach that handles the collision avoidance between agents and between structural obstacles[17]. In the model, each collision scenario is first analysed by examining the relative position of two colliding agents. Once the type of collision has been identified, an appropriate rule is then selected by the agent to avoid the collision. Lamarche derived a new mathematical model to compute the trajectories for individual pedestrians [68]. The model placed

emphasis on using path-finding algorithms to avoid structural obstacles using a novel method of dividing the environment into convex cells. Collision avoidance between agents are handled using a reactive steering algorithm.

Rymill focused on generating pedestrian trajectories by considering the mental states of individual agents during navigation [93]. In particular, Rymill proposed a realistic vision and attention model to model the sensory inputs of the agents. Feurtey’s model also performed different collision avoidance behaviours between pedestrians using a collision zone concept derived from the agent’s vision of the environment [27]. Shao proposed a rule based approach to identify the different possible types of inter-agent collision and then execute the corresponding subroutine to avoid the collision [98]. The model relies on subroutines like *step aside* and *full brake* commands to plot a collision free trajectory for the agent. As a result, Shao’s model does not actually take into consideration the mental states of the agent and may therefore not be effective in modelling the personalities of different pedestrians.

Due to the extensibility of an agent based model, we have used the agent based approach as the basic methodology for representing pedestrians within a crowd. Our approach is therefore to look at individuals within a crowd and try to fit a model to explain the various observed pedestrian phenomena. In particular, the physical constraints and the physiological states of a real pedestrian have been given focus. As compared to the approach adopted by Treuille in generating a crowd of homogeneous pedestrians [119], our goal here is to create a truly heterogeneous population capable of demonstrating interesting real-life social human behaviours for hundreds of different indi-

viduals.

2.5 Distributed Approaches

Due to the representation of individual agents within a pedestrian crowd, agent based simulations are inherently distributed. For a single core processor system, the simulation can be updated *sequentially*, i.e., each agent is moved and state of the environment updated before other agents are processed. Although, sequential processing of the agents may simplify many problems, the approach is physically unrealistic. Therefore a more realistic approach may be to use *synchronous cycles* to update the environment and move the agents within the simulation, i.e., agents are moved before the environment receive updates about the new position of the agents. The key challenges of using synchronous updates are the maintenance of spatial exclusivity between agents and the means to recover from any actual intersection between agents. The key benefits of using a distributed approach to modelling a pedestrian simulation include

1. Increased realism of the simulation;
2. Increased scalability of the simulation by allowing multi-processors to help compute the path of individual agents;
3. Increased behavioural capabilities of the agents due to increased processing power.

Lozano for example has proposed a scalable distributed system that handles not only the rendering of the crowd itself but the behaviour of indi-

viduals within the crowd as well [73]. The architecture also includes a load balancing module responsible for distributing the load of computing the simulation among different servers. In Lozano's model, an *Action Server* is used to maintain the states of the virtual environment which and maintain the adjacency information between each agent in the virtual environment. This information can then be queried by the agents themselves to achieve spatial awareness. The second key component is the *Client Processes*, which is each responsible for updating the states of a group of agents. The actual number of agents handled by each process is computed by the load balancing module. Navigation of each agent is achieved by applying the A* algorithm on a cellular automata divided grid. Collision detection between agents is an optional feature in Lozano's model.

In addition, there exist several works that targets on using distributed environments for autonomous agent simulations. For example, Kelin has experimented using High-Level Architecture (HLA) for traffic simulation [54]. In particular, Kelin outlined the advantages of using HLA as a means of allowing dynamic information to be integrated into the simulation during real-time. Logan explained the problem of using a distributed environment for the simulation of autonomous agents and proposed a new dynamic interest management approach for multi-agent simulation [72]. The model was tested using a simple prey predator scenario, to study the effect of different flocking behaviours on the population of preys and predators. However, the behaviours demonstrated in the test simulation are overly simplified and hence cannot be compared to requirements of a realistic pedestrian simulation.

Although agent-based models are inherently distributed, i.e., each agent is a fully self-sufficient unit by itself, a significant gap still exists between single processor based agents and agents designed to operate on a distributed system. One such method of interfacing agents with HLA has been described by Lees [70]. The method basically creates wrapper class to encapsulate basic agent level services from being aware of the presence of the HLA, i.e., the agent will not be aware that it is actually operating in a distributed environment. However, such an approach was found to have significant computational overheads over a single processor agent model. This suggests that the key advantage of using a distributed environment for a pedestrian simulation lies in its scalability property rather than the computational speed up of individual agents when the processor is still under-loaded.

Therefore, distributed simulation of pedestrian behaviours is especially useful for large scale crowd simulations. However the existence of significant challenges such as synchronous shared state access and the general difficulty of setup, have prevented full potential of a distributed system from being realised for a real-time pedestrian simulation.

2.6 Summary

Pedestrian simulation is used in many modern 3D computer graphics, games, evacuation scenarios and military simulations. For real-time applications, a balance between the level of detail of an entity's visual model, the richness of the entity's behaviour, and the computational costs of rendering these behaviours on screen, is required for the simulation to maintain an acceptable

frame rate.

As a result, we have discussed several methods of modelling a pedestrian crowd, and each methodology is found to be suitable for different applications. The flow based method provided a macro-level view of the environment, and it can be used for example, to identify key bottlenecks in the flow network. Cellular Automata can be used to simplify the environment and reduce the degrees of mobility of individual agents. Such an approach will allow studies to focus solely on the flow aspects of the simulation. Both the flow based approach and the cellular automata models by themselves are not capable of supporting a continuous levels of detail for the simulation. For example, collision avoidance between agents are not modelled in flow based approaches and discretely modelled in cellular automata models.

Particle Based systems represents each agent as a particle with self-driven capabilities. This approach is also able to distribute agents throughout the environment in a seemingly realistic manner. However, *pseudo-forces* are used to move the agents rather than forces that originate from the agents themselves, this is despite the fact that particle based models label autonomous agents as self-driven.

The Agent Based approach models each agent in the crowd as trackable individuals and usually after some real-life entity. The *potential field* approach is one of the more popular methods for modeling crowd behaviours due to its simplicity and extensibility. While this approach has the potential of controlling large crowds, several shortcomings of the approach have resulted in a variety of other agent-based models being proposed.

We have also discussed the use of distributed virtual environments (DVE)

for autonomous agent simulation. Key challenges for using DVE includes the general availability of the hardware and the complexity of different interest management schemes. However, DVE provides the opportunity to create not only a larger crowd but also a crowd with more complex individual and group behaviours.

After reviewing some of the different methodologies that can be used to model a pedestrian simulation, we have found the agent based method to be the most promising approach for our study. Agent based approach benefits from the fact that realistic human-like components such as sensory inputs and visual memory can be modelled for each individual. This allows the agent representation to be closer to that of a pedestrian. The level of detail used to model each individual can also be modified to adapt to the requirements of the application. For example, representation for autonomous agents can be enhanced to include attributes like age [69] and gender [79] to represent pedestrians with different physical capabilities and personalities. In the next section, the representation of an autonomous agent will be proposed. The agent will be modelled as closely as possible to a pedestrian within the constraints of a real-time simulation.

3 Representation of A Pedestrian

A pedestrian simulation needs to balance the visual details of individual agents, their behavioural models and the computational costs of using an agent model. Visual effects can be in the form of polygonal details for individual agents or the number of agents simulated in the virtual scene. Behavioural models dictate the relative behavioural complexity of each agent, i.e., they determine whether the agents behave like a simple program or like a real pedestrian. Our focus here is to create a representation for an autonomous agent that is

1. Realistic and accurate in the modelling of a real pedestrian;
2. Computationally efficient for use in a pedestrian crowd simulation;
3. Memory efficient for simulating crowd on a single workstation;
4. Extensible for customising the behaviour of the agent.

Based on these objectives, our requirements for a realistic representation of an agent for a pedestrian crowd include

1. A computationally efficient representation of the agent's in the virtual world for motion synthesis;
2. A computationally efficient but realistic model of the agent's sensory representation for receiving updates about the environment;
3. A reactive agent that is aware of both its environment and the agents around them;

4. A computationally efficient but realistic model of the agent's sensory and visual memory representation for modelling individual perception;
5. A computationally efficient but realistic model of the human attention system for improving the realism of agent representation.
6. A computationally efficient navigation model capable of describing realistic pedestrian behaviours and representing different pedestrian personalities.

In order to accurately model a pedestrian, we have proposed a two-level navigation mode; the macro-level and micro-level, to describe the navigation behaviours of real pedestrians. Macro-level navigation in our context is concerned with decisions such as finding the exits to a building without visually verifying the exit's condition. On the other hand, micro-level navigation is concerned with how to steer the agent to avoid a dynamic obstacle on a collision path. A two level navigation model is used because

1. Real pedestrians are goal-oriented and usually have some destination they want to reach;
2. Real pedestrians formulate a rough plan to their destination before they start moving towards their destination;
3. Real pedestrians rely on reflexes to avoid dynamic obstacles along the way to their destination;
4. Real pedestrians do not replan their rough path when a collision with another pedestrian is predicted.

Due to the importance of an agent's navigation model to the generation of realistic pedestrian behaviours, we will only discuss the micro-level navigation model in Section 4 and the macro-level navigation model in Section 5.

In this section, we will focus our discussion on the information gathering process of the agent from its surrounding. We start by discussing the various methods and considerations for modelling an agent after a real pedestrian in Section 3.1. Based on these considerations, we will then propose our agent's representation of a pedestrian in Section 3.2 and conclude our discussion on agent representation in Section 3.3.

3.1 Methods and Considerations of Agent Representation

In this section, we will focus on the methods and considerations necessary to derive a human-like model for the information gathering process of the agent, within the constraints of a real-time crowd simulation. The information gathering process of a pedestrian will typically include

1. Multiple sensory inputs for gathering raw information about the environment;
2. Built in awareness to recognise and filter raw information from the sensory;
3. Visual and Sensory memory to cache and store the information for further processing;

4. An attention system to target the sensory inputs to search for objects of interest.

The representation of the agent's body in the virtual environment is also equally important since it determines the computational complexity required to move the agent around. The body representation of the agent will also determine how collision between multiple agents will be resolved. For example, a complex body representation will require a physics solver to be implemented in order to demonstrate primary and secondary collision response for different objects.

Therefore it is important to balance the accuracy and realism of the representation for an agent with its computational costs to ensure that the simulation is able to run with a frame rate that is required by an application. For example, maintaining a frame rate of 24 frames per second is necessary for a smooth animation for most applications. To this end, it will be necessary to understand the different methods and considerations of representing an autonomous agent before proposing a formal representation.

3.1.1 Body Representation

In real-life, a pedestrian has a physical body that obeys the general rules of physics such as momentum and collision impulses. For example, a pedestrian will have a collision response when the pedestrian collides with different objects in the environment. The same physical body is also the body that is being detected or seen by other pedestrians. Therefore, the body representation of an agent in our context, is concerned with the necessary information

required represent the agent in the virtual world. The appropriate body representation is important because it

1. Determines the computational requirements for updating the agent's position in the virtual world;
2. Realism of the agent's collision response and it's resultant visual feedback of the collision;
3. Memory requirements needed to represent the agent's body in the virtual world.

There are three key methods of modelling the physical volume of human-like character [24]. The first method considers the body of the agent as a *point-mass*. The key advantage of using a point-mass representation is the approach's computational efficiency and lower memory requirements. For example, since torque has no effect on a point-mass, we do not have to consider rotational momentums on the body of an agent and therefore simplify the computations. The disadvantage of this approach is that point-masses produces less realistic collision response. For example, a shoulder to shoulder collision will not cause a visible deflection reaction from both agents.

The second method, commonly known as a *rigid-body* representation, considers the body of the agent as a rigid collection of points. Using this representation, a center of mass is first defined for the body with a finite volume. The *rotational degrees of freedom* of the body will then determine how the body will rotate in response to an external force that is applied on to the body. A virtual human model using a *rigid-body* representation will

hence be able to produce more realistic animations due to the addition of rotational momentum in the body's equations of motion. The obvious trade off in this case will be an increase in computation cost required to resolve the agent's rotational momentum.

The third method produces even more realistic animations by treating the agent's body as a network of *mass-spring*, also known as the *deformable-body* approach. This approach uses springs with a physical mass to model the shape of an agent's body. The general shape of the agent's body will then be maintained by the tensions within these springs, i.e., the springs will hold the shape of the agent's body together. This allows for real-time and dynamic deformations to be observed on the agent's 3D model. However using existing methods, the computational cost for propagating the forces and updating the states of individual springs in a deformable-body is still relatively high. As a result, it is currently not feasible to use deformable bodies to represent agents in a real-time crowd simulation.

3.1.2 Sensory Representation

Pedestrians sense other objects and agents in the environment as they are moving in order to avoid collision, recognise landmarks for navigation or simply to enjoy the sight itself. Therefore, sensory representation in our context refers to the front-end information gathering process of our agent. A typical sensory representation will consist of external sensors that pick up any object of interest from the environment for further processing by the agent. The representation of an agent's sensory input is necessary because it

1. Represents the first level of *awareness* about the virtual environment

for the agent;

2. Helps model the individual perception of an agent allowing for different personalities to be represented;
3. Determines the computational overhead required for the agent to gather information about the environment;
4. Realistically model the information gathering process of a real human.

However, not all agent representations will require a sensory representation, since it will depend on whether individual perception is required. For example in games, agents can be modelled to have omnipotent visual capabilities such as always knowing where the human player is located in the virtual world. Although this is sometimes termed as a 'cheating' agent, the computation savings from such an approach has made it a possible alternative to modelling individual perception for the agents.

Agent sensory inputs can be implemented using virtual sensors that mimic human-like sensory inputs like sight, sound and touch. These sensors are necessary for the agent to reproduce human-like navigation behaviours [114]. There are several existing attempts to create realistic sensory systems for autonomous agents based on observations about real humans. Herrero [45] for example proposed a realistic sensory model that considers the biological construct of a human's eyes and ears. Kim [56] examined how the human perception system automatically attunes itself to look for objects of interest and proposed a model to describe such a behaviour. Conde [19] proposed a vision filter that helps to clean up and combine the information gathered from an agent's virtual sensors for further processed.

An explicit representation of an agent's sensory inputs will allow us to realistically model the information acquisition process of a real human. For example, the agent will consider agents that are hidden behind walls to be invisible and therefore risks collision if the other agent suddenly appears. Based on these sensory inputs, the agent will then be able to react to different situations in the environment in a more human-like manner. At the same time, an explicit representation will also allow us to reproduce the behaviours of agents in different situations with diminished sensory abilities. For example, the modelling of evacuation route choice for agents under conditions of low visibility [52].

3.1.3 Environmental Awareness Representation

Pedestrians are expected to be aware of the environment they move in order to find their way around the environment. In particular, pedestrians know how to move through the environment even if the destination is not within the immediate sights of the pedestrian. This suggests that pedestrians have an abstract map of the environment and that pedestrians are expected aware of their relative position within this abstract map. A pedestrian may become lost if they do not have this abstract map or is unable to position themselves within the map, then they become lost. For example, it is common to see a child being lost in a shopping mall he or she is not familiar with. Environmental awareness in our context therefore refers to the agents' ability to recognise and find their way around the environment. Environmental awareness also includes the agent's ability to avoid structural obstacles as they move through the environment. Therefore the representation of environmen-

tal awareness is important because it

1. Represents how a real human recognise its relative position within the environment;
2. Represents how a real human knows the means to get to a particular location;
3. Determines the level of *awareness* and hence the computational complexity of our agents;
4. Determines the memory requirements needed to model individual perception about the environment.

However, environmental awareness does not cover an agent's ability to gather information about the environment, that is the responsibility of the agent's sensory inputs. This distinction is important because, different individuals are expected to rely on different sensory inputs to gather the necessary information required to gain environmental awareness. For example, some people may rely on the rhythms of a person's footstep to know who is approaching even without visual confirmation.

The level of environmental awareness required for the agent affects the computational complexity of the agent's information gathering process. In some cases for example, an agent may be required to recognise the presence of structural obstacles by analysing the geometry of the environment [127]. While in other cases, a navigation graph that has already marked out accessible and inaccessible regions of the environment may already be available to the agent. Therefore, the level of environmental awareness required will

affect the type of information to be gathered by the agent’s sensory inputs, and each type of information may require different computation complexity to gather. For example, recognising objects based on their geometry may be more computationally expensive than determining if a known object can be picked up by the agent’s sensory inputs.

There are two key approaches of enabling our agents to have environmental awareness. The first method allows the agent to gain environmental awareness by querying an abstract representation of the environment such as a scene graph [68], [77], [103], to obtain information about its relative position within the environment. The agent may also use the scene graph to detect structural obstacles like walls instead of obtaining this information from the environment’s geometry. The setbacks of using this approach include the need for additional memory requirements to store the representation, and the additional routines that are needed to construct such a representation.

The second method uses a direct detection of the environment, i.e., the agent relies on edges and contours of the objects to recognise and react to objects in the environment [127]. The direct detection approach may seem more natural and realistic than the use of an abstract representation such as a scene graph for achieving environmental awareness. In particular, when the agent is tasked with the exploration of an unknown area, the construction of an abstract representation of the environment will not have been possible. However, such the direct detection approach is also much more computationally expensive than the use of an abstract representation. Furthermore, our focus is not on intelligent object recognition but rather the generation of realistic navigation behaviours. Hence the direct detection approach is not

considered to be feasible for our pedestrian simulation.

3.1.4 Inter-agent Awareness Representation

Pedestrians need to be aware of the movements of other pedestrians around them so that they are able to make the appropriate responses to avoid collision. Pedestrians will also recognise their friends and kins and will behave differently around them. For example, kids will follow only their parents and not any other stranger, people will wait and help their own elderly parents or grandparents instead of overtaking them. Inter-agent awareness in our context therefore refers to an agent's ability to recognise and interact with other agents in the environment. Similar to environmental awareness, inter-agent awareness is not concerned with how the information about other agents are obtained, only on how these information are interpreted by the agent. Inter-agent awareness is also concerned with an agent's ability to avoid other agents during navigation. As a result, the representation of inter-agent awareness is important because it

1. Represents how real humans are aware of the existence of other humans around them;
2. Determines how they behave in different social context and around different people;
3. Forms the basis of group and kinship behaviours;
4. Determines the computational complexity required for the agents to gather information leading to inter-agent awareness;

5. Represents the memory costs of representing and storing these information required for inter-agent awareness;

In a real-time pedestrian simulation, it is not always necessary for agents to be mutually aware of each other. For example, if the camera is sufficiently far enough to make strict spatial exclusivity between agents a redundant feature then it may be possible for the agents to be represented without inter-agent awareness because the agents do not have to avoid other agents as they move around the environment. As a result, many existing macroscopic simulation of autonomous agents such as [2] and [8], have no built-in inter-agent awareness for their simulation.

On the other hand, if the simulation is interested in modelling how the behaviours of individual agents affect the behaviours of those around them, then it may be necessary to model this mutual awareness among the agents in the environment in order to reproduce realistic results. For example, fatalities during evacuation due to pedestrian flow bottlenecks as a result of poorly designed buildings [83], [100] can be realistically simulated only if the agents in the scene demonstrate the ability to wait and queue behind other agents in the environment. In order to do that, the agents must first be aware of the other agents before the appropriate behaviours can be executed. Therefore it is common for existing works on microscopic autonomous pedestrians such as [84], [97] and [112] to have some form of inter-agent awareness.

3.1.5 Sensory and Visual Memory Representation

Pedestrians are able to remember and recall the landmarks in the environment to aid them in their navigation. They are also capable of anticipating

the movements of other pedestrians around them based on what they have seen so far and use that information to avoid any potential collision with those pedestrians. *Sensory Memory* in our context therefore represents a cache of what has been sensed by the human sensory inputs after the original stimulus has disappeared. If the sensory memory contains information that the agent considers to be of interest, that piece of information will then be recorded into the agent's *visual memory* [5]. The representation of the agent's sensory and visual memory is important because it

1. More accurately represents the information gathering process of a real human;
2. Represents the combination of information from different sensory inputs for further processing;
3. Allows individual agents to have their own perceptive of the environment that can be influenced by different personalities and experiences;
4. Determines the memory requirements of the application for representing individual perception.

The agent's sensory and visual memory should not be confused with the agent's long and short term memories. Specifically, *sensory and visual memory* represents the *image* of what the agent has seen rather than the actual duration of that memory. For a more detailed understand of the human memory, refer to reference [5]. For our study, we are concerned mainly with the *sensory and visual memory* of an agent's perception about the environment and about the agents in the environment.

Therefore, this *perceived* state about the environment represents what that has been *remembered* by the agent rather than the actual state of the environment. The visual memory of the agent can be built for each agent as they explore the environment or *given* to the agent at the start of the simulation. Although the representation of individual sensory and visual memory will enable us to create a unique perception of the environment for each agent, this approach is not without its shortcomings. For example, each agent will now need to maintain a copy of the environment and this will result in a higher run-time memory requirement for the simulation. However individual perception is necessary to ensure a distributed approach for representing our agents. It is therefore important to balance the amount of information that is stored by each agent in the sensory and visual memory, in order to minimise the run-time memory requirements for representing individual perception.

3.1.6 Attention Representation

Pedestrians are sometimes seen to collide with a wall or other moving objects such as vehicles because they are not paying attention to their surroundings. For example, a pedestrian can be run over by a vehicle because both the driver and the pedestrian did not notice each other. Human attention can therefore be defined as a type of cognitive process where the human brain selectively concentrates on some points of interest in the environment. The representation of *attention* is important because it

1. Represents one of the basic but frequently overlooked function of the human brain when modelling autonomous agents;

2. More accurately model the information gathering process of a real human;
3. Describes how agents sensed by the agent are *selected* for *conscious* processing.

Posner pointed out three important functions of the human attention system [87], [88]. The human attention system is first used to orientate the sensory inputs to follow certain events. The sensory tracking of moving objects is achieved by *expecting* the next position of the object of interest based on the object's speed and other cues. If the prediction is wrong, humans will momentarily lose track of the object. Sensory tracking includes the control of both the direction and range of the sensory input.

The second use of the human attention is to actively scan for signals for *conscious* processing. This means that a pedestrian may not necessarily *notice* all the objects in the environment even if the object is *sensed* by the pedestrians sensory input. This distinction is important because *sensing* but not *noticing* allows for more realistic navigation behaviours to occur. For example, a pedestrian may be sensed by another pedestrian but because both pedestrians are engrossed in their own activity, they actually collided as a result of not paying attention.

The third use of the human attention system is for a pedestrian to maintain on a constant alert state. The state of *alert* depends upon the current action and intention of the human. For example, a human engrossed in reading will be much less vigilant than a police who is actively searching for evidence in a crime scene. It can be hence be argued that a human atten-

tion model is necessary for the agents to demonstrate realistic behaviours. However as with previous considerations, while human attention modelling adds cognitive capabilities to the agent, it also adds to the computational complexity of the simulation.

In the representation of our autonomous agent, we are only concerned with the second use of human attention, i.e., *conscious* processing of objects in the environment. We do not have sufficiently varied roles for our autonomous agents to consider the different states of *alertness*, although we will be able to extend our representation when the need arises. We are also less interested in the study of orientation targeting [18], [46], [56], because our agent are not expected to have that level of sensory precision.

Based on the considerations we have discussed thus far, we will describe our agent’s representation for a pedestrian in the next section. One of the key challenge here is the balance of model realism and accuracy with the memory requirements of the representation.

3.2 Agent Representation of a Pedestrian

A sufficiently accurate agent representation of a pedestrian is required for the demonstration of realistic pedestrian navigation behaviours. In this section, we will focus on describing such a representation and explain the rationale behind the choice of each form of representation.

We will first discuss our agent’s body representation in Section 3.2.1. The agent’s sensory inputs and its sensory memory will be discussed in Section 3.2.2 and Section 3.2.3 respectively. We will then outline the agent’s

attention model and discuss its importance in the representation of a human-like character in Section 3.2.4.

3.2.1 Body Representation

We have already discussed in Section 3.1.1 the different methods and approaches for representing the body of an autonomous agents. Based on those considerations, we are interested in finding a computationally efficient representation of the agent's in the virtual world for motion synthesis. Specifically, we have used a *point-mass* representation for the representation of our autonomous agents because

1. Realistic collision response is not the focus of our pedestrian simulation;
2. Low-polygon 3D models are more computationally efficient as compared to mass-spring representation;
3. Success of existing models [43], [68], [119] with using point-mass representation for crowd simulation.

The agent's body has therefore been represented using a low polygon (< 300 triangular polygons) 3D model of a human with key-framed animations such as walking and running. By representing the agent's body using a low polygon model, we are able to minimise the computational overhead for rendering the agent. Using a key-frame animation approach, we can generate a smooth animation by simply interpolating the agent's 3D model between key-frames. This approach is more computationally efficient than the using other animation techniques such as the skeletal animation technique [82].

Our autonomous agents are expected to move on a two dimensional terrain. However, 3D visualisation is used so as to provide a user with different perspectives of the simulation. If the agents are expected to move on an uneven terrain, then a height map [4], [35] can be used to ensure that the agent moves *on* the terrain. The agent's position in the virtual world can then be represented as a single point. Let the agent's position in the virtual world at time t be $\vec{p}(t)$. On a two dimensional plane, the agent's position can be defined by Equation 3.1 where $x(t)$ and $y(t)$ are the agent's position with respect to the virtual world's primary axis at time t .

$$\vec{p}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \quad (3.1)$$

The agent's velocity, $\vec{v}(t)$ in the virtual world can then be defined using the agent's speed, $s(t)$, and orientation, $\vec{D}(t)$, as shown in Equation 3.3. The agent's orientation is defined by Equation 3.2 where $\theta(t)$ is the angle which the agent is moving with respect to a fixed horizontal axis. Using a forward Euler's method the agent's position at time $t + \Delta t$ can then be found using Equation 3.4.

$$\vec{D}(t) = \begin{pmatrix} \cos \theta(t) \\ \sin \theta(t) \end{pmatrix}, \quad -\pi \leq \theta \leq \pi \quad (3.2)$$

$$\vec{v}(t) = s(t)\vec{D}(t) \quad (3.3)$$

$$\vec{p}(t + \Delta t) = \vec{p}(t) + \vec{v}(t + \Delta t)\Delta t \quad (3.4)$$

Personal space is a virtual region that surrounds each person where the person considers to be their domain or territory [39]. The personal space perceived by each person depends upon the person's culture and personal preferences. For example a person living in a city will be used to having smaller personal space as compared to a shepherd who is used to having wide open spaces. Entering the personal space of a stranger may be considered rude and undesirable in many cultures [9]. As a result, personal space affects how the real humans move around in the environment [3], [6]. Real humans for example are not expected to move past other humans shoulder to shoulder when there is ample space to move around. This is because humans will always try keep the integrity of their personal space when moving around the environment. Personal space is expected to expand or contract based on the situation an agent is in [7], [74]. For example, in a dense crowd simulation, personal space are expected to shrink in order for the agent to navigate through the dense crowd.

Due to the influence of personal space on a pedestrian's movement, it is therefore important to model the personal space of a pedestrian in our agent's representation. For the purpose of computational efficiency and ease of describing our agent model, we have used a circular representation of an agent's personal space, centered at $\vec{p}(t)$ with radius r around the agent. Several existing models [68], [98], have also successfully used a circular personal space to influence an autonomous agent's navigation behaviour. However, if required by the application, a study can be made to examine the effects of different personal space representation on a pedestrian's behaviours. Changing the representation of personal space can be made with appropriate modifi-

Figure 3.1: Agent representation.

cations to our collision avoidance model. Figure 3.1 shows the point-mass representation of our agent together with the agent's position, velocity and personal space.

3.2.2 Sensory Representation

In order for an agent to be able to react to changes in the environment, the agent will need to collect updates about the virtual environment. Therefore in this section, we are interested in finding a computationally efficient but realistic model of the agent's sensory representation for receiving updates about the environment. The agent's sensory inputs are used by the agent to

1. Recognise different objects and agents in the environment;
2. Serve as an early collision warning with other objects in the environment;
3. Recognise its own relative position within the environment for the purpose of path-planning.

In order to equip our agents with human-like sensory capability, we modeled three sensory inputs, comprising of sight, sound and touch, for our agents. The agent's sight can be customized using two parameters, the agent's vision strength, d_v , and the agent's field of view, fov . The agent's vision strength represents how far the agent wants or is able to see. The absolute limit of the agent's vision strength will of course be the agent's own physical limits. However, a pedestrian will not always use their maximum visual strength during navigation. Instead, their vision strength will be influenced by their habits and situational needs. For example, if a pedestrian is attempting to cross a wide road, the pedestrian will need to scan further for on-coming vehicles before crossing. As compared to a pedestrian who is crossing a back alley will not be expected to look far and wide for on-coming vehicles.

The agent's field of view represents how wide the agent wants or is able to see. Modelling an agent's field of view allows us to represent the *blind spots* of a real human, i.e., real humans do not have a 360 degrees view of the environment. The entire range of the agent's vision field can therefore be represented as a circular segment centered at $\vec{p}(t)$.

Figure 3.2 shows a graphical representation of an agent's vision cone. We can also observe from the figure there are two methods of detecting objects within the vision field. The first method uses active scanning by dividing the vision cone into regularly spaced rays. The second method uses passive scanning whereby objects will notify the agent when they are within range of the vision cone. We have used active scanning for our vision model as we believe because it better represents the way real humans gather information

Figure 3.2: Agent vision model.

about the environment.

Let N_ν be the total number of divisions in the vision cone, then the total number of sampled rays will be $N_\nu + 1$. The i^{th} sampled ray of the vision cone can then be written as

$$\vec{r}_i(t) = \vec{p}(t) + d_\nu \begin{pmatrix} \cos\left(\theta(t) + \frac{i}{N_\nu}\nu\right) \\ \sin\left(\theta(t) + \frac{i}{N_\nu}\nu\right) \end{pmatrix}, \quad 0 \leq i \leq N_\nu \quad (3.5)$$

The number of divisions, N_ν , required to sample the space represented by the vision cone will depend on several factors. We will use Figure 3.3 to discuss these factors. Let $d_{\nu i}$ be different possible visual strengths of the agent such that $d_{\nu i} \leq d_{\nu(i+1)}$, and d_{ai} is the perpendicular distance between the limits of the vision cone such that $d_{ai} < d_{a(i+1)}$. The *Vision Density*, $\rho_{\nu i}$, for each $d_{\nu i}$ is then given by Equation 3.6.

Figure 3.3: Vision range and sample density.

$$\rho_{vi} = \frac{N_v - 1}{d_{ai}}, \quad N_v \geq 2 \quad (3.6)$$

Hence from equation 3.6, the value of ρ_{vi} decreases with d_{ai} and increases with N_v . Since d_{ai} is proportional to d_{vi} it can be concluded that vision density falls when the vision range of the agent increases and N_v remains unchanged. From Figure 3.3, it can also be observed that in order for all the objects to be detected by the agent, it is necessary for $\frac{1}{\rho_{vi}}$ to be greater than the diameter of the smallest object in the scene. Such an approach however increases the number of vision samples required for each agent and greatly increases the computational complexity of the simulation.

We therefore control the agent's vision density based on the estimated density of agents around the agent. Let ρ_{pop} represents the estimated population around the agent and $k_{\rho_{pop}}$ represents relative importance of ρ_{pop} on the number of divisions required. Also let, N_{min} and N_{max} is the minimum number and maximum number of divisions allowed respectively. Then Equa-

tion 3.7 defines the value of number of divisions, N_ν , required for the current situation.

$$N_\nu = N_{min} + k_{\rho_{pop}} \rho_{pop}, N_{min} \leq N_\nu \leq N_{max} \quad (3.7)$$

Once the number of samples have been determined, the parametric equations describing each ray sample is computed. The sample rays are then used to test for intersections against the bounding volumes of objects in the environment. The objects intersecting the rays are then sorted according to the distance from the $\vec{p}(t)$ to the point of intersection. If all objects in the scene are opaque then only the nearest object is considered to be detected by the sampling ray. This object will then be added to the list of visible objects for further processing. Duplicated objects that are sampled by different rays will not be added to the list of visible objects.

The agent's audio senses can modelled as a sphere around the agent with a range of d_ζ and centered at $\vec{p}(t)$. Although more accurate models exist [45], we have approximated the agent's audile capabilities using a sphere due to a sphere's computational and representation efficiency.

The agent's touch senses is modelled as a tightly packed bounding volume around the agent. At the same time, the bounding volume will also be used as an approximation for the agent's 3D model for the purpose of collision detection. Specifically, we have used an Axis-Aligned Bounding Box (AABB) [25] to represent the agent's touch sense. The choice of bounding volume is based on the following considerations

1. Memory efficiency in the bounding volume's representation;

2. Computational efficiency for an intersection test between a ray and the bounding volume;
3. Computational efficiency for updating the position of the bounding volume as the agent move through the environment.
4. Provide a sufficiently tight-fitting approximation of the agent's 3D model.

In particular, computationally efficient intersection tests between a ray and the choice of bounding volume is an important consideration because rays are used as our agent's vision samples. This means that intersection tests between the volume and a ray will be one of the most frequently executed routine. As a result, the intersection test must be extremely efficient in order to minimise the computation overhead for finding the list of visible objects for the agent. Based on these considerations, the AABB was found to be the most suitable form of bounding volume to represent the approximate shape of the agent's 3D model.

3.2.3 Sensory and Visual Memory Representation

The modelling of individual perception about the environment is important in our pedestrian simulation to create a truly heterogenous pedestrian crowd. Therefore in this section, we will describe a computationally efficient but realistic model of the agent's sensory and visual memory representation to model individual perception.

In real-life, pedestrians sense the environment through their sensory inputs which will first deposit raw information in the sensory memory. Only

information that are of significance will then be stored in the a pedestrian's visual memory. For example, dynamic information about a stranger in the streets is likely be discarded as soon as the stranger is seen. By comparison, an important landmark necessary for the pedestrian to locate its way around the environment will likely leave a visual imprint on the pedestrian even after the pedestrian has left the area.

Therefore in our model, information gathered by the agent about dynamic objects in the scene are stored for as long as the objects are of significance to the agent. For example, information that will help the agent navigate around the environment such as landmarks and exit locations will be stored in the agent's *visual memory*. The agent's visual memory can then be *recalled* when necessary for the agent to find its way around the environment.

In order to best represent the sensory memory of an agent. We have to introduce the concept of a *relative frame*. We define the *source agent* to be the agent currently scanning the environment for objects and *target agents* to be autonomous agents that are visible to the source agent. The relative frame therefore aims to provide a mathematical framework upon which the motion of the source agent relative to those around it can be better analyzed, [63]. A relative frame will hence exist between the source agent and each visible target agent. These relative frames will therefore form the *perceived* state of environment for the source agent. The relative frame is an important component for the our agents to achieve individual perception for achieving realistic navigation behaviours.

Figure 3.4 show a relative frame between a source and a target agent.

Figure 3.4: A relative frame.

The relative frame between the source and target agent at time t is composed of three components; the relative position of the two agents, $\vec{p}_r(t)$, the relative velocity of the two agents, $\vec{v}_r(t)$, and the combined personal space of the two agents, $r_r(t)$. The relative position between the source and the target agent, $\vec{p}_r(t)$ is defined by Equation 3.8 where $\vec{p}_s(t)$ and $\vec{p}_g(t)$ are the positions of the source and target agents respectively. Similarly the relative velocity is defined by Equation 3.9 and the combined personal space is defined by Equation 3.10.

$$\vec{p}_r(t) = \vec{p}_s(t) - \vec{p}_g(t) \quad (3.8)$$

$$\vec{v}_r(t) = \vec{v}_s(t) + \vec{v}_g(t) \quad (3.9)$$

$$r_r = r_s + r_g \quad (3.10)$$

The representation for the agent’s visual memory depends upon the purpose of the simulation. We proposed that visual memory should be used to represent the environment so that agents are able to gain environmental awareness. In this case, the actual representation of the visual memory will depend upon the representation used for the environment. Therefore our representation of the agent’s visual memory will only be discussed in Section 5.3 where the implementation of our pedestrian simulation is discussed.

3.2.4 Attention Representation

In order to better represent our agents as real humans, the modelling of an attention system is required. In this section, we will proposed a computationally efficient but realistic model of the human attention system for improving the realism of agent representation. Specifically, we will attempt to model the selective processing functionality of a human attention system, as we have discussed in Section 3.1.6.

Therefore, to represent that real humans do not always notice everything they see, we propose that each agent should have a specific number of *attention points* depending on the agents roles and actions. Agents should only take *notice* of objects with significant interest to the agent itself; for example objects that are on a collision course with the agent. At the same time, there is a limit to the number of objects that a real human can be attentive to simultaneously. Hence each time humans who takes *notice* of an

object, they will be less attentive of other objects in the environment. Using the same argument, agents engrossed in some actions may not take *notice* of the events that has occurred around them [100]. For example, a researcher engrossed with his or her own work may not notice of a fight going on just outside the lab.

We therefore attempt model a human's limited attention by subtracting a specific number of attention points from the agent's attention system each time the agent takes notice of an object in the environment. Once this pool of attention points has been exhausted, all other objects or events in the environment around the agent will be actively ignored until some attention points can be reclaimed. Attention points can be reclaimed either when the agent is interrupted from the current task or when the object of interest leaves the attention of the agent.

3.3 Summary

We have stated the objectives, requirements and considerations necessary for the representation of an autonomous agent in a crowd simulation. Specifically, the primary concern is to find a balance between the physical realism of the representation and the computational cost of using such a representation. To this end we have represented an autonomous agent by its

1. position, $\vec{p}$, relative to some global point of origin;
2. speed, s , where $s \geq 0$;
3. orientation, θ_i where $0 \leq \theta_i \leq 2\pi$ and relative to a fixed horizontal line;

Figure 3.5: Representation of an autonomous agent.

4. vision strength, d_v , where $d_v \geq 0$;
5. field of view, fov , where $0 \leq fov \leq 2\pi$ and relative to a fixed horizontal line;
6. circular personal space with radius, r , and center $\vec{p}$;
7. list of N agents visible to the agent, represented by $A = [A_1, A_2, \dots, A_N]$.

In addition, the agent's sensory memory is composed of multiple relative frames that make up the agent's perceived state of the environment. The source agent's, A_s , relative frame with a visible target agent, A_g is then composed of the

1. relative position, $\vec{p}_r$, where $\vec{p}_r = \vec{p}_s - \vec{p}_g$;
2. relative velocity, $\vec{v}_r = s_r \vec{D}(\theta_r)$, where s_r is the relative speed and θ_r is the Euler angle of the relative velocity;

3. relative personal space with radius, r_r , and center, $\vec{p}_r$, where $r_r = r_s + r_g$.

The agent's body is also represented as a low polygon 3D model and approximated using an AABB, which in turn is also used for collision detection (Figure 3.5). The white bounding box around the agent is an axis-aligned bounding box that is used for collision detection tests. The blue ring around the agent is the agent's personal space which is used for calculating navigation parameters for the agent. Green lines originating from the agent are the agent's visual samples. The length of the green lines represents the agent's visual range, d_v , and the angle between the two extreme visual samples represents the agent's vision angle, fov .

We have therefore demonstrated how a pedestrian can be represented as an agent in our pedestrian simulation. However, we have not addressed how an agent is able to move through the environment in a realistic manner. This is achieved using our micro-level and macro-level navigation model. In particular, we will focus on how the micro-level navigation model can be used to steer the agent to avoid collision with other agents in the next section. We will also describe how different constraints can be added to the basic micro-level navigation model to simulate different pedestrian personalities and physical restrictions.

4 Micro-Level Navigation Behaviour for Pedestrians

The representation of an agent of a pedestrian has been discussed in the previous section. In this section, the focus will be on how to move the agent in a realistic manner that mimics the navigation behaviours of a pedestrian. The proposed model also has to be computationally and memory efficient for use in a medium sized crowd.

Figure 4.1 shows the proposed model of an autonomous agent representing a pedestrian. The agent's sensory representation models a pedestrian's primary sensory inputs as we have described in Section 3.2.2. The sensory estimation error attempts to model human errors made during the estimate of parameters such as the object's distance when sensing the object using the agent's sensory inputs. Different pedestrians are expected to have different amount of estimation errors based on their habits and experiences. The agent's attention system is used to represent a pedestrian's selective processing of objects in the environment as we have discussed in Section 3.2.4. The agent's memory representation attempts models a pedestrian's sensory and visual memory as we have discussed in Section 3.2.3.

In order to accurately model the navigation behaviours of a pedestrian, we have used a two-level navigation model, the *macro-level navigation module* and the *micro-level navigation module*. A two-level navigation model is used because

1. Pedestrians are goal-oriented and usually have some destination they

Figure 4.1: Proposed representation of an autonomous pedestrian.

want to reach;

2. Pedestrians formulate a rough plan to their destination before they start moving towards their destination;
3. Pedestrians rely on simple movements like side-stepping to avoid dynamic obstacles along the way to their destination;
4. Pedestrians do not replan their rough path when a collision with another pedestrian is predicted.

A similar two-level approach has been proposed by Noser [81], Lamarche [68] and Musse [80] to model an agent's navigation capabilities. Noser's work represents one of the earliest attempts to create an autonomous agent capable of moving in a virtual environment. Many important agent representations such as visual sampling of the environment by an agent are still modeled in recent works on micro-level navigation.

The difference between Lamarche's model and our own is that our micro-level navigation model also considers the personalities and physical constraints of the agent when computing the steering parameters for the agent. Musse used a procedural approach to generate micro-level navigation behaviours. However, Musse does not take the personal preferences of individual agents into consideration. For example, a pedestrian may not always want to avoid another pedestrian on a collision course. By comparison, our model parameterizes these preferences and considers them together when computing the agent's next step.

Work related to micro-level navigation includes the study of collision avoidance between pedestrians [27] and the modeling of collision avoidance

behaviour using potential fields [50], [92]. The potential field method achieves collision avoidance by introducing a distance based repelling force between the agents. However, this approach is not intuitive and does little in explaining the choice of route of individual agents. Goldstein uses the relative orientation between the agent and static obstacles to calculate the attraction and repulsion forces used to steer the headings of an agent [32]. Goldsteins model does not consider relative velocity which is important for interactions between dynamic obstacles. By comparison our model is able to provide a unified relationship between the relative velocity and orientation required for collision avoidance. Finally, rule based methods such as [93] and [98], attempt to identify different types of possible encounters between agents and then attach a rule to define how the agent should then move to avoid a collision. The problem with this approach is that the rules derived frequently include the habits and experiences of how the researcher perceive to react in such an encounter. By comparison, our model allows the customisation of the agent's navigation behaviour to reflect a different cultures and social norms.

4.1 Micro-Level Navigation Model

As shown in Figure 4.1, the macro-level navigation module will supply the micro-level navigation model with a rough plan for the agent to reach a destination. This is represented as a sequence of waypoints tracing from the agent's current position to the destination. The micro-level navigation model is then responsible for getting the agent from one waypoint to another while

avoiding collision with other agents in the environment. The micro-level navigation model has therefore the following requirements:

1. A method of representing the steering parameters for our agents to generate realistic navigation behaviours.
2. A computationally efficient collision prediction mechanism capable of providing sufficient warnings before the actual collision;
3. A computationally efficient method of computing the steering parameters for collision avoidance;
4. A method of combining different physical constraints and personality considerations to form the navigational behaviours for our agents;
5. A realistic method of resolving conflicting needs and constraints similar to real pedestrians.

In order to satisfy these requirements, we have proposed a new micro-level navigation model capable of describing the realistic navigation behaviours of our agents. The model basically considers the navigation of our agents to be consisted of a number of constraints, where each constraint describes a restriction on the movement of agent in order to satisfy the constraint. The following types of constraints have been defined for our agent:

1. Avoidance constraints to describe the agent's need to avoid a single agent in the environment;
2. Speed Constraints that can be used to describe for example, the agent's physical limits, tolerance to speed changes of other agents and urgency to reach the destination;

3. Orientation Constraints that can be used to describe for example, the agent's physical limits and the agent's current course.

The use of constraints instead of relying on a rule-based approach [27], [78], [98], because new constraints can be easily added to the agent's navigation model without affect existing constraints. At the same time, each constraint can be highly specialised and targetted at describing a need or requirement of a pedestrian. For example, multiple speed constraints can be used to describe a pedestrian's maximum attainable speed and comfortable walking speed together.

The micro-level navigation model consists of the following steps. First the agent will need to stay on alert and predict any potential collision with another agent. Collision prediction between the agent and other moving agents will be discussed in Section 4.1.1. Once a collision has been predicted, the agent will need to change its speed and orientation in order to avoid other agents in a way it is most comfortable with. Section 4.1.2 will explain how this can be achieved. As it has been discussed that different agents may have different speed constraints that is physical or situational in nature. Section 4.1.3 will explain how the speed constraint can be formulated and combined with any collision avoidance constraint that the agent may already have. Similarly, the agent may have orientation constraints that can be used to keep the agent on course towards its destination. This will be discussed in Section 4.1.4. Then in Section 4.1.5, the method of combining the different constraints together will be explained. Section 4.1.6 will explain why the presence of conflicting constraints is important and how they can be resolved in the model. We will then close our discussion about the micro-level

navigation model in Section 4.2.

4.1.1 Collision Prediction

Pedestrians are intelligent beings that will try to avoid a collision before they actually occur. At the same time, pedestrians tend to avoid collision only when they are close to the point of collision [14]. In this section, we will explain how collision can be predicted and how this information can be represented by the agent to determine the appropriate time to avoid the collision.

The first step towards predicting a collision is to identify the *collision zone* between the source and target agent using the *relative frame* described in Section 3.2.3. The collision zone is defined as a region of space where the source agent should avoid in order to avoid a predicted collision between the source and target agent. Note that the collision zone between the two agents depends upon the relative position of the two agents and is independent of their relative velocity. Therefore, a reasonable strategy to predicting a collision will be to observe the relatively velocity between the source and target agent.

Let $\sigma_r(t)$ be the Euler angle of the relative position, $\vec{p}_r(t)$, then to properly define the collision zone, we can introduce two parameters, $\theta_{r\min}$ and $\theta_{r\max}$ such that

$$\theta_{r\min} = \sigma_r(t) - \sigma(t) \quad (4.1)$$

$$\theta_{r\max} = \sigma_r(t) + \sigma(t) \quad (4.2)$$

Figure 4.2: Collision zone between the source and target agent.

Assuming the two target agents do not change their velocities, then $\sigma_r(t)$ will not change even as the two agents get closer together. On the other hand, $\sigma(t)$ will increase as the two agents get closer together, enlarging the collision zone between the two agents. This means that it may be easier to avoid a collision that is further away than to avoid one that is nearer. By defining the relative position, $\vec{p}_r(t)$ of the source and target agents in terms of their relative distance apart, $d_r(t)$ and its associated Euler angle, $\sigma_r(t)$ in Equation 4.3, we can find the value of $\sigma(t)$ using Equation 4.4.

$$\vec{p}_r(t) = d_r(t) \begin{pmatrix} \cos\theta_r(t) \\ \sin\theta_r(t) \end{pmatrix} \quad (4.3)$$

$$\sigma(t) = \arcsin\left(\frac{r_r(t)}{d_r(t)}\right) \quad (4.4)$$

Having found the collision zone, the next step will be to find the condition for collision to occur between the source and target agent. Referring to Figure 4.2, it can be observed that collision may occur if the relative velocity of the source and target agents, $\vec{v}_r(t)$, is within the collision zone. Let $s_r(t)$ and $\theta_r(t)$ be the speed and corresponding Euler angle of the $\vec{v}_r(t)$, then $\vec{v}_r(t)$ can be written as:

$$\vec{v}_r(t) = s_r(t) \begin{pmatrix} \cos\gamma(t) \\ \sin\gamma(t) \end{pmatrix} \quad (4.5)$$

Based on these observations, a collision is predicted to occur if both agents maintain their current course and if $\theta_r(t)$ lies within the range defined by Condition 4.6. In order for the source agent to avoid the collision, the source

agent will hence need to change its current headings so that the relative velocity between the source and the target agent is no longer within the collision zone. Of course if no collisions are predicted by the source agent then the source agent is expected to continue with its current headings.

$$\exists \theta_r(t) \{ \theta_r(t) : \theta_{r \min} \leq \theta_r(t) \leq \theta_{r \max} \} \quad (4.6)$$

Although the same observation has been made by Feurtey [27], Feurtey did not analyse the information using a relative frame. Instead the collision zone was used as a heuristic cost function to steer the agent away from the collision zone. By comparison, we represented the collision zone within the relative frame, and as a result, we can observe a whole new level of information that can be used to describe different navigational behaviours. At the same time, our model has the flexibility of allowing agents to decide on whether they will want to avoid a target agent as well as the means to avoid it. Figure 4.3 illustrates a predicted collision if Condition 4.6 is not satisfied. i.e., the relative velocity between the source and target agent continues to be contained within the collision zone of the two agents.

4.1.2 Collision Avoidance Constraint

Once a collision has been predicted between two pedestrians, each pedestrian may then choose the a time to avoid the other pedestrian or to ignore collision avoidance completely. The latter case is a common behaviour during panic situations and when the pedestrian expects the other pedestrian to give way. In this section, we assume that the pedestrian will want to avoid the other

Figure 4.3: Collision prediction.

pedestrian to prevent a collision and will describe the constraints on the agent's steering parameters necessary to achieve collision avoidance.

Specifically, we will describe a set of constraints on the source agent for the source agent to avoid a *single* target agent. We will describe how the constraint can be extended to avoid multiple target agents in Section 6. The target agent that is selected from the source agent's list of visible agents based on the collision risk it pose to the source agent. We will now discuss how the steering parameters of the source agent can be changed so as to avoid a potential collision. Our strategy for the source agent to avoid

collision with other agents in event of a predicted collision is as follows,

1. Select a candidate $\theta_r(t + \Delta t)$ such that $\vec{v}_r(t + \Delta t)$ does not lie within the collision zone between the source and target agent;
2. Change the source agent's speed, $s_s(t + \Delta t)$, and orientation $\theta_s(t + \Delta t)$ to achieve the candidate $\theta_r(t + \Delta t)$;
3. If no solution exists for $s_s(t + \Delta t)$ and $\theta_r(t + \Delta t)$, then a new candidate $\theta_r(t + \Delta t)$ has to be selected to avoid the collision.

Based on the observation that collision can be avoided by ensuring the relative velocity between the source and target agent lies outside the collision zone, the candidate $\theta_r(t + \Delta t)$ should be chosen such that

$$\neg (\theta_{r \min} \leq \theta_r(t) \leq \theta_{r \max}) \quad (4.7)$$

The choice of $\theta_r(t + \Delta t)$ is called the *collision avoidance strategy* of the source agent. In this thesis, we have randomly chosen $\theta_r(t + \Delta t)$ to be either $\theta_{r \min}$ or $\theta_{r \max}$. It is also possible to select a $\theta_r(t + \Delta t)$ based on empirical studies about the pedestrian [14];

However there exist more than one combination of $s_s(t + \Delta t)$ and $\theta_r(t + \Delta t)$ to achieve the candidate $\theta_r(t + \Delta t)$. This observation can best be explained using Figure 4.4. The primary reason for this is that the relative speed between the two agents does not play any role in helping the source agent avoid a collision with the target agent. Hence there is no constraint on what the final relative speed between the two agents should be. However this observation naturally reflects the navigational choices available to

Figure 4.4: Possible new velocities of the source agent.

a pedestrian. That is, a pedestrian can choose to avoid other pedestrians by changing the speed or orientation or a combination of both. Hence the primary challenge here is to compute the new $s_s(t + \Delta t)$ and $\theta_r(t + \Delta t)$ that takes into consideration the *habits and personality* of the source agent.

To address this challenge, a relationship between the agent's speed and orientation has been derived. This relation is called the *duality property* and is given by the following equation:

$$s_s \sin(\theta_s - \theta_r) = s_g \sin(\theta_g - \theta_r) \quad (4.8)$$

where $s_s(t + \Delta t)$, $\theta_s(t + \Delta t)$, $s_g(t)$ and $\theta_g(t)$ is the speed and orientation of the source and target agent respectively. $\theta_r(t + \Delta t)$ is the candidate Euler angle of the relative velocity between the source and target agent that is required to avoid the collision. The Duality Property naturally reflects how people will try to avoid collision in daily life. For example, some people may attempt to steer clear of a collision by changing their orientation while

Figure 4.5: The Duality Property.

minimising changes to their speed. Other people may use short bursts of speed but minimise their deviation from the current path to avoid collision. As illustrated in Figure 4.5, the duality property can be easily proven.

To determine the desired speed and orientation for the source agent to avoid the target agent, we have assumed that pedestrians prefer to minimise changes in their orientation when avoiding obstacles. In order to find the minimal change in orientation, we need to derive the constraints on the source agent's steering parameters in terms of $\theta_s(t + \Delta t)$. However, this assumption is for the purpose of describing our approach to computing the agent's steering parameters. If instead, it is required to model pedestrians who prefer to minimise changes to their speed, then the derived constraints should be written in terms of $s_s(t + \Delta t)$.

Given the candidate $\theta_r(t + \Delta t)$ necessary for the source agent to avoid a target agent, we need to first find the required change in the source agent's orientation. Then the new speed of the agent can be found using the duality property as shown in Equation 4.8. In order to minimise the required change

in orientation for the source agent to avoid the target agent, let

$$\omega_s(t + \Delta t) = \theta_s(t + \Delta t) - \theta_s(t) \quad (4.9)$$

Then we will have the following collision avoidance constraint (l_1) on the source agent's $\theta_s(t + \Delta t)$:

If $\sin(\theta_g - \theta_r) < 0$,

$$(k_1 < \omega_s < k_1 + \pi) \wedge (k_2 - \pi < \omega_s < k_2) \quad (4.10)$$

otherwise,

$$(k_1 - \pi < \omega_s < k_1) \wedge (k_2 < \omega_s < k_2 + \pi) \quad (4.11)$$

where $k_1 = \theta_s(t + \Delta t) - \theta_s(t)$ and $k_2 = \theta_g(t) - \theta_s(t)$. $l_1 \wedge l_2$. In the case of $\sin(\theta_g - \theta_r) = 0$, then the velocity of the target agent is collinear with the relative velocity between the source and target agent. This is a trivial case and will not be discussed in this thesis. We will now give the proof for l_1 .

Proof:

$$\vec{v}_s - \vec{v}_g = \vec{v}_r$$

$$\therefore s_s \sin \theta_s - s_g \sin \theta_g = s_r \sin \theta_r$$

Using the duality property,

$$\frac{s_g \sin(s_g - \theta_r) \sin \theta_s}{\sin(\theta_s - \theta_r)} = s_r \sin \theta_r \quad (4.12)$$

The left hand side of Equation 4.12 can then be simplified as,

$$\begin{aligned}
 LHS &= \frac{\sin \theta_s \sin \theta_g \cos \theta_r - \sin \theta_s \cos \theta_g \sin \theta_r}{\sin \theta_s \cos \theta_r - \cos \theta_s \sin \theta_r} - \sin \theta_g \\
 &= \frac{\cos \theta_s \sin \theta_g \sin \theta_r - \sin \theta_s \cos \theta_g \sin \theta_r}{\sin \theta_s \cos \theta_r - \cos \theta_s \sin \theta_r} \\
 &= \frac{\cos \theta_s \sin \theta_g - \sin \theta_s \cos \theta_g}{\sin \theta_s \cos \theta_r - \cos \theta_s \sin \theta_r} \sin \theta_r \\
 &= -\frac{\sin(\theta_s - \theta_g)}{\sin(\theta_s - \theta_r)} \sin \theta_r
 \end{aligned}$$

$$\therefore \frac{\sin(\theta_s - \theta_g)}{\sin(\theta_g - \theta_r)} = \frac{s_r}{s_g}$$

Since $\frac{s_r}{s_g} > 0$,

$$\sin(\theta_s - \theta_g) \sin(\theta_s - \theta_r) < 0$$

Using the duality property,

$$\frac{s_g}{s_r} \sin(\theta_s - \theta_g) \sin(\theta_g - \theta_r) < 0$$

$$\therefore \sin(\theta_s - \theta_g) \sin(\theta_g - \theta_r) < 0 \quad (4.13)$$

Using the same argument, then according to Equation 4.8, both $\sin(\theta_s(t + \Delta t) - \theta_r(t + \Delta t))$ and $\sin(\theta_g(t) - \theta_r(t + \Delta t))$ must have the same sign. That is,

$$\sin(\theta_s - \theta_r) \sin(\theta_g - \theta_r) > 0 \quad (4.14)$$

Then from conditions 4.13 and 4.14, we can then see that if $\sin(\theta_g - \theta_r) < 0$,

$$\sin(\theta_s - \theta_g) > 0 \wedge \sin(\theta_s - \theta_r) < 0 \quad (4.15)$$

otherwise if $\sin(\theta_g - \theta_r) > 0$,

$$\sin(\theta_s - \theta_g) < 0 \wedge \sin(\theta_s - \theta_r) > 0 \quad (4.16)$$

By solving conditions 4.15 and 4.14 in terms of ω_s , it will then be trivial to show that l_1 is true.

In addition to the collision avoidance constraint, a real pedestrian may have their own comfortable walking speed and destinations to reach. As a result, having collision avoidance constraints alone may not be sufficient to describe a very rich set of pedestrian navigation behaviours. Therefore, more constraints on the agent's steering parameters are needed to generate more realistic navigation behaviours.

4.1.3 Speed Constraint

In real-life, pedestrians are expected to move at different speeds when navigation through the environment [69]. These differences can be due to personal preferences like the pedestrian's own comfortable walking speed, physical limits like the pedestrian's maximum attainable speed, or situational requirements like the pedestrian being already late for work. Based on this observation, we introduce the concept of speed constraints to our micro-level navigation module. In our context, we define speed constraints as the range of speed that the agent is willing or able to move when navigating through the environment. Speed constraints for the source agent can be simply defined as

$$s_{min} \leq s_s \leq s_{max} \quad (4.17)$$

where s_{min} and s_{max} is the minimum and maximum speed allowed by the source agent. The limits as we have discussed can be used to represent the agent's current mental state such as being patient (low minimum speed) or impatient (high minimum speed). For example, by modifying the value of s_{max} , it is possible to model the maximum attainable speed and comfortable walking speed for different pedestrians. Similarly, by increasing the value of s_{min} , it is possible to model the urgency of an agent to reach a destination or its patience to wait for slower moving agents.

However, the form described by Equation 4.17 cannot be combined with the agent's collision avoidance constraint which is in terms of ω_s . We will therefore have to rewrite the speed constraint for the source agent to be in terms of ω_s . The *Duality Property* can be used in this case to describe $s_s(t+\Delta t)$ in terms of $\theta_s(t+\Delta t)$. The agent's speed constraint l_2 can therefore be written as:

If $\sin(\theta_g - \theta_r) > 0$,

$$\frac{k_3}{s_{min}} \leq k_4 \leq \frac{k_3}{s_{max}} \quad (4.18)$$

otherwise,

$$\frac{k_3}{s_{max}} \leq k_4 \leq \frac{k_3}{s_{min}} \quad (4.19)$$

where $k_3 = s_g(t) \sin(\theta_g(t) - \theta_r(t + \Delta t))$ and $k_4 = \sin(\theta_s(t + \Delta t) - \omega_s - \theta_r(t))$. Note that the agent's speed constraint is still not entirely in terms of ω_s although it can be easily found using simple trigonometric identities. Interested readers may refer to Appendix 8.2 for the proof of Constraint l_2 and the specific constraints in terms of ω_s for different values of k_3 .

4.1.4 Orientation Constraint

Orientation constraints represents a pedestrian's desire to stay on course in order to reach a location in the environment. In addition, orientation constraints can also be used to describe a pedestrian's tolerance to changes in orientation during navigation. An orientation constraint, l_3 , on the source agent's orientation, $\theta_s(t + \Delta t)$ can be defined as

$$\theta_{s \min} \leq \theta_s \leq \theta_{s \max} \quad (4.20)$$

where $\theta_{s \min}$ and $\theta_{s \max}$ represents the agent's physical or desired limits on its orientation. Orientation constraints can be used to keep the agent in a lane by limiting the values of $\theta_{\min}$ and $\theta_{\max}$ to represent the limits of the lane the agent is in. An orientation constraint can also be used to represent the agent's reluctance to change its orientation to avoid other agents by constraining ω_s to a small range of values.

Figure 4.6 illustrates how orientation constraint can be used to move the agent towards the exit of the room. Note that $\theta_{s \min}$ and $\theta_{s \max}$ take into consideration both the personal space of the agent and the physical opening of the exit, to ensure that there is sufficient physical clearance for the agent to reach the exit.

4.1.5 Combining Constraints

We have described several constraints on the source agent in terms of $\omega_s(t + \Delta t)$. In this section, we shall describe the method of combining the constraints in order to compute the new steering parameters for the source agent

Figure 4.6: Use of an orientation constraint.

that considers all the constraints. The first step to combining the constraints for the source agent will be to normalize all the constraints on $\omega_s(t + \Delta t)$ to the space defined by Equation 4.21. Therefore it may be necessary to wrap some of the constraints around π and $-\pi$ during the normalising process. This means that the original constraint on $\omega_s(t + \Delta t)$ may in some cases be normalized into a union of two or more constraints. Figure 4.7 shows a normalisation process where $x_1 < \omega_s < x_2$ and $x_2 > \pi$. The dotted line represents the original constraint and the solid line represents the final normalised constraint. After the normalisation process, the original constraint becomes a union of two normalised constraints such that $-\pi \leq \theta_s \leq \pi$.

$$-\pi \leq \theta_s \leq \pi \quad (4.21)$$

After normalization, constraints can then be combined by performing an intersection between other normalized constraints. Figure 4.8 outlines how

Figure 4.7: Normalising the constraints.

Figure 4.8: Combining different constraints.

different constraints can be normalized and combined together. Visually, the method can be described using a spanning tree where the number of levels depends upon the number of unique constraints, C_i , in the system. When a branch generates a ϕ set, that branch is terminated without considering further constraints. The valid solutions of $\omega_s(t + \Delta t)$ is then defined by all the non-empty leaf nodes of the spanning tree. As a result, the range of $\omega_s(t + \Delta t)$ described by the solution set will be able to describe a realistic navigation path for the source agent who have considered the respective constraints.

```

Function Range IntersectRange(Range a, Range b){
    Range result;
    if(a.lower > b.higher || a.higher < b.lower)
        return NULL;

    if(a.lower < b.lower)
        result.lower = b.lower;
    else
        result.lower = a.lower;

    if(a.higher > b.higher)
        result.higher = b.higher;
    else
        result.higher = a.higher;

    return result;
}

```

Figure 4.9: Pseudocode for computing the constraint intersection.

Figure 4.9 shows the pseudocode for the intersection test between two normalized constraints. The algorithm first attempts to quickly reject the two constraints by testing if the constraints intersect. If the constraints do not intersect then an empty set is returned and the branch terminates. However if the constraints do intersect, then the overlapping region is returned and the test continues until there are no more constraints.

The final step will be to find a value of $\omega_s(t + \Delta t)$ from the set of valid values of $\omega_s(t + \Delta t)$. Since we have assumed that the agent will attempt to avoid any collision with the least possible deviations from their current headings, then ideally $\omega_s(t + \Delta t) = 0$. However if $\omega_s(t + \Delta t) = 0$ does not lie within the solution set then the next closest value of $\omega_s(t + \Delta t)$ will be

used. Let a valid range of $\omega_s(t + \Delta t)$ be defined as

$$\omega_{s \min} \leq \omega_s \leq \omega_{s \max} \quad (4.22)$$

where $-\pi \leq \omega_{s \min} \leq \pi$, $-\pi \leq \omega_{s \max} \leq \pi$, and $\omega_{s \min} \leq \omega_{s \max}$. Then the method of choosing $\omega_s(t + \Delta t)$ can therefore best be described as

$$\omega_s = \begin{cases} 0, & \text{when } \omega_{s \min} \leq 0, b \geq \omega_{s \max} \\ a, & \text{when } \omega_{s \min} \geq 0 \\ b, & \text{when } \omega_{s \max} \leq 0 \end{cases} \quad (4.23)$$

Having found a $\omega_s(t + \Delta t)$, the new orientation of the source agent, $\theta_s(t + \Delta t)$, can then be found using Equation 4.9. Similarly, given the new orientation of the source agent, the new speed of the source agent, $s_s(t + \Delta t)$, can be computed using Equation 4.8. The new steering parameters can then be used to move the agent forward in time using the forward Euler method.

4.1.6 Resolving Conflicting Constraints

We have introduced three types of constraints; collision, speed and orientation constraints, that helps control the movement of an agent in the virtual environment. In particular, these constraints allow the agent to mimic the navigational behaviour of a real pedestrian. We have also outlined the method of combining the different constraints together in order to compute the new steering parameters for the source agent have realistic navigation behaviours. However it is inevitable that the constraints will result in an

Figure 4.10: Conflicting constraints.

empty solution set for the source agent due to conflicting constraints. Figure 4.10 shows an example scenario, where the new velocity of the source agent $\vec{v}_s(t + \Delta t)$ based on the orientation constraint of the source agent to stay on course will result a on a collision course with the source agent. In this case, the agent will not be able to satisfy both the collision constraint and the orientation constraint.

Therefore we will now discuss how conflicting constraints can be resolved by our micro-level navigation module and explain why collisions can potentially be a realistic behaviour for pedestrian simulations. If we are to take a step back and analyse the scenario in Figure 4.10, we can reason that there are actually three potential methods of resolving the conflict.

The source agent can choose to observe the collision and speed constraint and ignore the orientation constraint, in this case the agent will produce an

Figure 4.11: Prioritising constraints.

overtake maneuver. The source agent can also choose to observe the collision and orientation constraint, in this case the agent will slow down behind the target agent until either the collision threat is gone. Finally the agent can choose to ignore the collision constraint and enforce the speed and orientation constraint. The result is the source agent will collide with the target agent and maybe knock the target aside.

This means that it is necessary to prioritise the constraints of the agent according to the mental states of the agent [15]. However our model does not

practice mutually exclusive priority, i.e., constraints with a lower priority are not skipped when a higher priority constraint has been observed. Figure 4.11 outlines how different constraints are organized in our model to reflect the priorities an agent have during navigation. A constraint higher on the priority queue will be combined first before those constraints with a lower priority. If the newly combined constraint creates an empty set, the constraint is discarded and the changes rolled back. The algorithm terminates where there are no more constraints on the priority queue.

4.2 Summary

We have therefore described the entire micro-level navigational model for our autonomous pedestrians. The model organizes and combines different constraints to compute the steering parameters for the next time step based on the agent's mental state and sensory memory. In particular, three main types of constraints have been discussed;

1. Collision constraint attempts to steer the agent away from collision with a target agent;
 - (a) *Duality Property* is used to compute the new steering parameters for the source agent to avoid a collision;
 - (b) Specific avoidance rules govern the special case where the relative velocity of the source and target agent is collinear with the velocity of the target agent.
2. Speed constraint attempts to steer the source agent in such a way that

Figure 4.12: Micro-level navigation model.

the physical or desired speed constraints of the source agent is observed;

3. Orientation constraint attempts to steer the source agent in such a way that physical or desired orientation of the source agent is observed.

Figure 4.12 summarised the proposed micro-level navigation model. The agent will first gather information about the environment through the agent's sensory input. The micro-level navigation model will then predict collision threats and select an avoidance strategy in the form of $\theta_r(t + \Delta t)$. Based on the selected strategy, the collision avoidance constraint on the source agent can then be formulated. Speed and orientation constraints can be added to generate more realistic navigation behaviours that takes into consideration the agent's personality, habits and physical limits. These constraints can then be combined together to find the agent's new orientation. The Duality Property can then be used to compute the agent's new speed. We will also demonstrate how the micro-level navigation model can be used to gener-

ate basic and self-organising navigation behaviours in Sections 7.3 and 6.2.1 respectively.

There are however two major setbacks with the micro-level navigation model. The first problem is the model only consider one collision constraint each time the steering parameters are computed, i.e., only the target agent with the highest risk of collision is considered for avoidance. For a sparse crowd, this approach is sufficient, however for a dense crowd, the results are less realistic. We shall discuss the solution to this problem in Section 6.

Secondly, the micro-level navigational model does not have the capability of steering the agent through a complex scene. A macro level navigational model is required to aid the agent in planning a path especially when the agent's goal is not within the agent's line of sight. Therefore in Section 5, we shall describe how a macro level navigational model can be used to add path-planning capabilities to an our agent.

5 Macro Level Navigation Behaviour for Pedestrians

We have discussed the micro-level navigation model for our agent in the previous section. However, the micro-level navigation model alone is not enough for the agents to move around in a complex environment. This is because the agent may not necessarily have visual affirmation of the destination in a complex environment. For example, the exit to a building may be several floors below or visually obstructed behind a wall. As a result, it will not be enough for the agent to move in the general direction of the exit because those routes may be blocked by structural obstacles such as walls.

The macro-level navigation model therefore aims at enabling agents to use an abstract representation of the environment to plan a path to its destination. The path should also be influenced by macro-level factors such as distance to the destination and the traffic conditions of the route. Plans can be made with the help of influence maps [12],[76], graph search techniques [103], or constraint based planning [121] over a proper representation of the environment. In particular, we have used the A* algorithm as our primary path planner [40].

Referring to Figure 4.1, the macro-level navigation model composes of two key components; the agent's visual memory and path planner. The visual memory represents an abstract map of the environment that is known to the agent. The path planner will use the agent's visual memory to plan the rough path for the agent to move from one point of the scene to another. In addition, the destination of the agent will depend upon the goals and needs

of the agent. Note that the path-planner is only responsible for planning a *rough plan* for the agent, the actual execution of this path will be done by the micro-level navigation model. This is because it is not realistic for a pedestrian to replan its path in the event of a predicted collision. This approach is similar to actual human navigation behaviour [33] and has also been used in famous works of Musse [80] and Lamarche [68]. Therefore, the objectives of the macro-level navigation module are to

1. Build an abstract map of the environment for the agent to gain environmental awareness;
2. Allow the agent to know its relative position within the environment;
3. Allow the agent to plan a rough path to reach a location in the environment;
4. Allow the agent to receive and use spatial information about the environment such as traffic conditions to influence the agent's rough plan.

For a real-time pedestrian crowd simulation, we are concerned with the macro-level navigation module's

1. Memory requirements for
 - (a) Storing the representation of the environment for modelling individual perception about the environment;
 - (b) Running the algorithm to prune the environment representation for feasible routes.

2. Computational efficiency for real-time path planning by multiple individuals within the crowd.
3. Ability to generate realistic pedestrian behaviours for the agents;
4. Ability to integrate dynamic information during real-time to influence the path choice of the agents.

We will first discuss the various methods of path finding for pedestrian simulation in Section 5.1. In Section 5.2, we will describe our proposed macro-level navigation model. Then in Section 5.3, we will discuss the role of the agent's visual memory in generating realistic navigation behaviours. Finally we will close our discussion about macro level navigation in Section 5.4.

5.1 Methods and Considerations for Path-Finding

There are two major considerations when designing the path planner for the agent;

1. Representation of the environment for path-finding;
2. Algorithm used for path-finding.

Representation of the environment for path finding focuses on representing information about the environment for strategic and tactical assessment. The method of representing the environment affects the quality of the path produced by the planner which in turn affects the realism of the pedestrian behaviours as we describe in Section 5.1.1. In addition, different methods

of representing the environment also has different memory requirements to store the representation. This is an important consideration if each agent is to have a copy of the environment to represent the different knowledge of the environment held by different individuals in a pedestrian crowd.

The second consideration is for algorithm used in the path-planner. The concern here is also the memory needed run the algorithm as well as the general computational efficiency of the algorithm. For example graph based representation of the environment can use optimal graph search techniques like Dijkstra or heuristic graph search techniques like A*. Optimal graph search techniques explores all possible alternatives to produce the best possible path for the agent. By comparison, heuristic graph search techniques estimates the best path using heuristics and continues to explore that path until the goal is reached or when a dead-end is encountered. As a result, heuristic approaches do not produce the best possible paths for the agent but computes faster than optimal graph-search approaches. We will describe three main classes of path-finders in Section 5.1.2.

In Section 5.1.3, we will propose our macro-level navigation model based on these considerations. In particular, we will discuss how we formulated the heuristics for the A* algorithm to better reflect the mental states of a pedestrian during path-finding.

5.1.1 Representation of Environment for Path-Finding

The representation of the environment for path-finding basically attempts to simplify the environment for the purpose of representing the areas that are accessible to the agent for efficient path-finding. That is the representation

focuses on the connectivity between different areas of the environment and describes how a person can get from one point of the environment to another without passing through structural obstacles such as walls. The appropriate representation of the environment is important because it

1. Determines the quality of the path produced;
2. Determines the memory requirements needed to store the representation of the environment, especially if individual perception of the environment is required;
3. Determines the computational efficiency of finding a path using the representation for the agent;
4. Determines if the representation can be constructed automatically.

The quality of a path is an relative measurement used to describe how natural the path generated by the path-planning algorithm looks when compared to the path a pedestrian will take in a similar scenario. For example, the generated path will not look natural if there are many abrupt changes in direction when the environment is just a simple corridor.

Secondly, different environments may need varying memory requirements using different representations. For example, if the environment is represented using points, then a large number of points will be needed to sufficiently represent the space that an agent can move in. As compared to a representation using volumes such as spheres, fewer volumes are needed to cover the entire environment. Having more points also translates to having a larger search space for a suitable route to the destination. This in turn trans-

lates to higher computational cost to plan a new route due to the increased time needed to evaluate all the options. Therefore having efficient representation of the environment for path-finding will help ensure lower memory requirements and a faster simulation.

Finally, the ease of constructing the representation for a new environment should also be considered. For example, complex schemes of representing the environment may not readily have a method of automatic construction for the representation. For a complex environment, significant manpower will then be needed to transform the environment into a representation suitable for path-finding. These manpower costs can then add up significantly for each change to the environment. It is therefore important to balance ensure that the representation can be automatically created using existing methods.

In this section, we shall describe three main types of representations; *point based representation*, *volume based representation*, and *hierarchical representation*. We will evaluate how each of these representations fare against our considerations on memory and computational efficiency.

Point Based Representation: The point based approach represents the environment using *waypoints*. The waypoints are positioned such that they represent points in the environment that are accessible to the agent. These waypoints are then joint together to represent the possible routes from one location of the environment to another that is free from structural obstacles. Tozour [118] proposed a corner graph where waypoints are placed on the convex hull of an obstacle so that agents are limited to walking along the

edges of the convex hull. A similar approach has been proposed by Habib called the MAKLINK, where the midpoints of a convex open spaces is used as waypoints for intelligent agents to follow and avoid structural obstacles [37]. Additional information can also be placed into waypoints graphs to create human-like movements [116], tactical movements [71] and strategic planning [77]. As seen from Figure 5.1, much of the empty space in the environment cannot be reached by an agent using a point based representation of the environment. In order for the agent to be able to used access those empty spaces, additional nodes have to be added into the graph. The realism of the agent's movement in the environment therefore depends on how dense the nodes are placed in a given area. On the other hand, a dense waypoint graph increases the memory requirements for storing the representation and increases the search space for finding a route through the environment. Therefore point based representation is less suitable for representing wide open spaces than volume based representations. In particular, point based representations will require a large number of nodes to achieve quality paths for the agents.

Volume Based Representation: Volume based representation of the environment attempts to represent the environment using convex volumes such that the space enclosed by each volume is completely free of structural obstacles. For example path-finding using *Cellular Automata* [10], [31], will require the environment to be overlaid using a set of regular sized grids. Each cell on the grid will represent a specific volume of space rather than a single point. *Circle shaped waypoints* [118], attempts to find the largest circle that can be fitted into an open space without intersecting any structural ob-

Figure 5.1: A corner graph for path-finding.

stacles. Transversal between waypoints are allowed only if the circles of both waypoints intersect each other. Circle-based waypoints has lower memory requirements than point based representations because entire volumes can be represented using only the circle's center and its radius. As a result, they are better suited to represent wide open areas than point based representations. However, circle-based waypoints are not tight-fitting approximations of the environment, e.g., empty spaces near sharp corners cannot be completely covered by the circular waypoint as shown in Figure 5.2. Another type of volume based representation is the *Navigational mesh* [103], [123], which represents the environment using polygonal meshes. The key advantage of using this approach is that navigational meshes can potentially represent all types of environment, both indoors and outdoors. They are also generic enough to

Figure 5.2: A circle graph for path-finding.

prevent excessive subdivision for oddly sized obstacles. However polygonal mesh requires more memory to represent than the other methods described so far because the mesh is basically represented as a list of triangles. Furthermore, the construction of a navigation mesh for navigation is non-trivial. One such method of constructing a navigation mesh has been proposed by Pettré [85].

Hierarchical Based Representation: Hierarchical based representation of the environment for path-finding, subdivides the environment recursively until the contents are homogeneous, e.g., all the children contain only accessible space for the agent to move in. The construction can also be terminated

when a predetermined depth for the representation has been reached. For example, the *Quadtree* [23], [96], recursively subdivides the environment into four such that its leaf node consist entirely of structure obstacles or free space. Quadtree is useful for representing regions in the environment that are accessible to the agent moving on a two dimensional plane. Another well known hierarchical based representation is the *Portal engine*. A *Portal engine* is an innovation that originates from the game *Duke Nukem 3D* by *3D Realms* [24]. A portal engine basically divides the scene into convex sections and organise the sections together as a flat graph. *Portals* are then added to any edge where two or more convex sections touches. In doing so, these convex regions are now connected through the *portals*, where the *portals* themselves forms a higher level representation of the environment. Each of these convex regions can then be tessellated using either point based or volume based representations of the environment for the purpose of path-finding. Specialized hierarchical representations can also be derived for the purpose of partial planning. For example, Sturtevant proposed a hierarchical waypoint structure to be used together with the proposed partial-refinement A* (PRA*) algorithm [108].

Using the proposed representation and PRA*, partial plans that contains a combination of low and high level waypoints can be generated. Low level way points are then given to the agent for immediate execution. On the other hand, high level waypoints help to ensure that the final path travelled by the agent is still near optimum. Hence hierarchical representations of the environment allows for optimization of the path-finding algorithm by reducing the number of nodes that a path-finder must visit in order to plan

a new path for the agent. However, it may be possible that the memory requirements of a hierarchical representation to be higher than a purely point or volume based representation [106].

We have thus looked at three main types of representation. In particular, we have adopted a hierarchical based approach to improve the computational efficiency of finding a path through the environment. A hierarchical based approach reduces the time needed to query if a region of space contains an obstacle. Unlike a point based or volume based approach, the query can quickly focus on the region to test for obstacles. Furthermore, a hierarchical based approach allows different information to be stored at different levels of the graph. For example, individual areas representing a restaurant can be collectively grouped together so that the agent will be able to locate the relative position of the restaurant without having to search through an entire list.

In the next section, we will examine the different algorithms that can be used for path-finding and evaluate their memory requirements and computational efficiency for use in a pedestrian simulation.

5.1.2 Methods for Path-Finding

Algorithms for path-finding can be broadly classified into three types; influence mapping, graph based searches and path tables. For a pedestrian crowd simulation, it is important to use the appropriate algorithm for path-finding because it

1. Determines the memory requirements required to run the path-planning algorithm;

2. Determines the computational costs for path replans;
3. Determines the planner's ability to integrate dynamic influences in the planning of a path.

Some algorithms may require a specific representation of the environment to function while others may require special data structures to support its search for a suitable path. As a result, different algorithms may need different memory requirements, which for a real-time pedestrian crowd simulation is an important consideration. It is therefore important to understand the memory requirements of each algorithm and how it grows or shrinks during the execution of the algorithm.

Search based algorithms will understandably perform slower than look-up based algorithms. This will in turn impact on the computational costs for path replans. In particular, in a dense pedestrian crowd simulation, consecutive collision avoidances between agents may steer the agent far off course requiring relatively more frequent path replans than in other scenarios. Therefore the computational costs of finding a path for an agent is an important consideration for the simulation to run with a reasonable frame rate.

Another important consideration is the planner's ability to integrate dynamic influences in the planning of a path such as local traffic conditions. Look-up algorithms may for example be unable to do so because the paths are all pre-computed before the execution of the simulation. As a result real-time information like traffic congestions cannot be used by an agent to plan for detours. In a pedestrian simulation, the inclusion of dynamic information such

as traffic congestions is important to generate realistic pedestrian navigation behaviours. For example, it will not be realistic for all the agents to crowd at one exit of the building when obviously there is a less crowded one nearby.

In this section, we will therefore evaluate the three classes of path-finding algorithms based on its memory requirements and computational efficiency. We will also consider the algorithm's ability to integrate local influences such as traffic information into the path-planning process.

Influence Mapping: Influence mapping was introduced by Tozour as a method of representing how an agent perceive other objects and agents in 2D planar scene [117]. Influence mapping first divides the scene up into regularly sized grids. Each grid cell then has an attractive or repulsive force that depends upon the cell's content to pull or push the agent towards or away from the cell. Therefore, influence maps are in particular useful for strategic and tactical assessments. For example, Pottinger proposed that influence mapping can be used to determine the placement of resource sites in a real-time strategy game [89]. Kent proposed that multiple influence maps can combined together for intelligent tactical group movements and terrain analysis [55]. Use of influence mapping for path finding [110], and for defense building by [34] has also been proposed. Figure 5.3 shows how influence mapping can be used to affect the planned path for an autonomous agent who wants to avoid the crowd. In this case, the influence map records the population density of the region the cell represents. The density value can then be used to push the agent's path away from the congestion such that the resultant path of the agent is one that appears to have intelligently

Figure 5.3: Congestion avoidance.

avoided the congestion.

Waypoints and Graph Search: By representing the environment as a graph, path-finding can be achieved by using graph search techniques. The Dijkstra's algorithm is perhaps one of the best known algorithm used to find the shortest path between all pairs of nodes [22] . However Dijkstra's algorithm produces the optimum path using an exhaustive search and is therefore computationally expensive to search for a route in a complex environment. Another comparable algorithm will be the Floyd-Warshall algorithm which is capable of handling negative weights in the connections between different nodes [28]. For real-time applications, the A* [40] algorithm is a more popular approach used for path-finding due to its faster average run-time. The A* algorithm basically uses heuristics to estimate the cost to the destination and does not perform an exhaustive search for the path, i.e., based on the heuristics, the algorithm will terminate once a path has been found, it will not attempt to verify if the path is indeed the most optimal. However, the

A* algorithm requires the agents to have complete knowledge of the environment, i.e., all parts of the environment must be connected in order for the A* algorithm to work. Stentz therefore proposed the D* algorithm to handle situations where priori information about the environment is either incomplete or completely absent [104]. For example, a pedestrian may not know that the parts of the current route are now closed for renovation and therefore inaccessible. As a result, the pedestrian may have to perform path-replans to find a new route only upon reaching the renovation site itself. While this behaviour is human-like and that it explains why notices for closure are usually placed earlier along the path, path-replans can be computationally expensive to perform. In particular, if full path replans when done repeatedly in a pedestrian crowd simulation, they can cause a serious deterioration in the performance of the simulation. Instead of using a complete path replan, D* by comparison is able to achieve better performances than A* by utilising information from previous planning episodes to plan for the current path. This reduces the need to replan the entire route starting from the agent's position to the destination. More recently, Koenig proposed the D* Lite algorithm based on D* algorithm which basically performs the same operations but is relatively easier to understand [61], [62]. Another path-finding algorithm was proposed by Korf, called the Learning Real-time A* (LSTA*) [66]. The algorithm is useful for agent based approach to modelling autonomous agents because LSTA* allows the agent to update its own copy of the environment in real-time. Basically LSTA* starts off with rough estimates for the cost of travelling between different points in the environment. As the agent experiences the environment, more accurate heuristics are then used to correct

the abstract representation of the environment. LSTA* will then take into consideration these new heuristics when planning for a new path.

Preprocessed Path Tables: Preprocessed path tables pre-computes and store the routes between all pairs of nodes in a graph representation of the environment. The method was first introduced by Surasmith for fast open terrain navigation [109]. Basically an optimal path planner such as Dijkstra’s algorithm is first used to find all-pairs shortest path off-line. The routes are then recorded in the form of a look-up table for retrieval during run-time. This means that the agent does not have to search for the path during run time, i.e., the runtime complexity of a preprocessed path table is $O(1)$. However the memory requirements of the path table is $O(n^2)$ where n is the number of nodes in the graph representation of the environment. Hierarchical representation of the environment has been proposed Dickheiser [21], and Sterren [106], to reduce the memory footprint of the path table. Dickheiser proposed a cluster grouping of nodes and store the path table only for these cluster nodes. In this way not only will the size of the path tables be reduced, the search space required path finding within the cluster is also reduced. Sterren proposed to combine path tables with portal engines so that nodes that represents inaccessible areas can be dropped from the path table. By comparison, a classical path-table will still record a node that is inaccessible with an infinite distance between all other nodes. Sterren’s approach therefore reduces the size of the path-table by removing inaccessible nodes from the table itself. Sterren also stated that although nodes cannot be added or removed from the table without reconstructing the entire

path-table, the path table can be *patched* during run-time if prior knowledge about how the environment will change is available. Patching is achieved by replacing part of the path table affected by the change with new routes that reflect the new changes.

We have thus discussed the three key approaches for path-planing in this section. In this thesis, we have used A* algorithm for the flexibility of its heuristics to reflect the navigation choices of a pedestrian. For example, some pedestrians may choose a route that is most comfortable to them while others may choose a route that is shortest. Our micro-level navigation model requires only the location of the next sub-goal as the output from the macro-level navigation model. Hence it is possible to substitute A* with other path-finding algorithms according to the needs of the application so long as it is able to output the next sub-goal that is within the line of sight of the agent. In the next section, we will describe some of the more specific macro-level path-planning considerations for a pedestrian simulation.

5.1.3 Considerations for Pedestrian Simulation

We have described the different types of environment representation for path-finding and discussed the various algorithms that can be used for path-finding. In this section, we will describe five additional macro-level considerations that are important in a pedestrian simulation; partial knowledge modelling, open-goal modelling, dynamic environment modelling, memory requirements, and execution interleaving. The first three considerations are important because they affect the realism, runtime computational complexity and the memory requirements of the pedestrian simulation. The fourth

consideration attempts to explain why memory is required for different representations and algorithms for path finding. Execution interleaving explains how path-planning can be distributed between multiple frames to help ensure a constant frame rate for the simulation.

Partial or Complete Information: Although in real-life, pedestrians are expected to have only a partial set of information about the environment they are moving in, representing partial information for an agent means that each agent will need to maintain an instance of the environment. For a pedestrian crowd simulation, this can add up to be a significant memory requirement for the simulation. A pedestrian's partial knowledge about the environment can be modelled using uncertainty and ignorance [102], [86]. Ignorance is a measure of how much information the agent has as compared to how much information are there in the environment. Uncertainty on the other hand is a measure of how strong one's belief is in the accuracy of the information already known [91]. This means that an agent can be ignorant and uncertain at the same time. When an agent is ignorant, it means that the agent only has part of the environment's abstract map, e.g., the agent does not always know the shortest possible route to the destination. When an agent is uncertain, it means that the agent does not know for sure the existence of a shortest path, e.g., the agent may try to explore the environment to find the shortest route.

Open-Goal Pathfinding: Pedestrians are expected to have multiple needs at any one time such as being hungry and late for work. As a result, pedes-

trians may have several destinations or sub-goals which they want to travel to in order to satisfy their needs. Open-goal pathfinding is therefore an attempt to model this behaviour in pedestrians. In particular, agents should have multiple means of achieving their goal and will have select one destination to for execution. For example, an agent is capable of selecting from several exits of a building to actually exit from. While open-goal path-finding may improve the realism of the simulation, open-goal path-finding can add to the computational complexity of finding a path for an agent to move through the environment. In particular, open-goal pathfinding means that means that multiple paths have to be evaluated based a variety of considerations like the difficulty of getting there (e.g., congestion) or proximity of threats (e.g., fire). If the goals are far apart from each other, then using a path-finding algorithm like A* will be computationally expensive due to the extended search space. A cheaper approach might be to use a distance table or influence map to estimate the cost of moving to different destinations. Once the goal with the lowest heuristic has been decided upon, the path finder can then be used to plot a route between the agent's current position and the goal.

Static or Dynamic Connections: In real-life, pedestrians move in an environment with closeable passageways and collapsible walls. Therefore to realistically model the environment for pedestrian simulation, the general connectivity of the environment should be made dynamic, i.e., connections between different areas are not constants. However, dynamic connections can have varying degrees of computational complexity and memory requirements depending on the approach used. For example, if a Quadtree is used to rep-

represent the environment for path-finding, then changes in the topology of the environment may require the Quadtree to be reconstructed. Similarly, if a preprocessed path table is used for path-planning, then changes in the connectivity within the environment will cause some routes to become invalid. When that happens parts of the table may need to be patched. The extent of the change will depend on whether the area changed is a major crossroad several different routes.

Memory Requirements: The memory requirement for a path-finder is an important consideration for a pedestrian crowd simulation because memory is a shared and limited resource that is required to ensure the continual working of the simulation. There are two types of memory requirements in a path-finder; the *storage memory* requirement of the representation and the *execution memory* requirement of the planning algorithm. *Storage memory* refers to the memory needed to represent the environment for path-finding, e.g., the nodes and edges of Quadtrees and waypoint graphs. *Execution memory* refers to the memory needed for the path-planner to work with, e.g., marking visited nodes and unvisited nodes using an A* path-planner. Graph based search approaches may have a higher execution memory requirement but a lower storage memory requirement than path table approaches. This is because graph based search algorithms, like A*, creates a spanning tree during execution time and the size of the spanning tree depends upon the complexity of the environment. The cost of representing the spanning tree during execution of the A* algorithm can be significant. There are two well known variations of A* that uses less memory than A* during the exe-

cution of the algorithm. They are the Iterative Deepening A* (IDA*) [65], and the Simplified Memory-Bounded A* (SMA*) algorithm [77]. On the other hand, path tables have a higher storage memory requirement but negligible execution requirement as compared to graph based planners. This is because path tables need to represent the shortest possible routes between all points in the environment, but agents merely need to look up the path table to find the shortest path required to reach any point in the environment.

Execution Interleaving: Execution interleaving refers to the deliberate pausing of a routine in order to service another routine before returning to complete the execution of the original routine. Execution interleaving is an important consideration in real-time pedestrian simulation because, path planners may need a relatively long time to plan the paths for all the agents in a complex environment. This means that execution interleaving will allow the simulation to run with a relatively constant frame rate and produce relatively smooth animations. Although path planners such as A* allows interleaving between execution as long as the open and closed list of the A* algorithm is preserved, the agent may have already moved between the execution of the planner therefore invalidating the plan. As a result, the agent will need to replan the path again on the next execution of the path planner. In this case, planners that are able to use partial plans, such as D* or D* Lite can be useful to avoid having to start over [61], [62]. In contrast to interleaving during a path-planning routine, a better approach may be to divide the agents into buckets, ensuring that each bucket of agents can have their path planning completed within the same frame.

In this section, we have therefore discussed the additional macro-level considerations for our macro-level navigation model. Based on these considerations we can now continue to discuss on our macro-level navigation model in the next section. The key concern here is the balance between the accuracy and realism of the model with the computational cost and memory requirements for using the model.

5.2 Macro-Level Navigation Model

Pedestrians are able to plan a route through the environment in order to reach their destination even when the destination is not within the sights of the pedestrian. At the same time, once a high level plan has been formulated, a pedestrian does not replan the path just to avoid another pedestrian. Instead a pedestrian will steer aside to avoid the pedestrian and then steer back on course to reach the destination. The macro-level navigation model therefore functions as the path-planning module for a pedestrian in that it plans a rough path for an agent to follow. The micro-level navigation model as we have discussed in Section 4 then fills in the role of steering the pedestrian to avoid other pedestrians. In this section, we will focus on describing our macro-level navigation model based on the considerations we have discussed in Section 5.1

Figure 5.4 shows the information flow for our proposed macro-level navigation module. First the polygonal representation of the environment is simplified using a Quadtree and tessellated with waypoints to form a waypoint mesh. An instance of this mesh is then loaded into an agent's visual

Figure 5.4: Macro level navigation model.

memory to represent individual knowledge about the environment. The A* path-finding algorithm is then used to plan a rough path for the agent to follow in order to reach a particular destination.

In this section, we will describe our proposed representation of the environment for path-finding in Section 5.2.1 and the heuristics used for the planner in Section 5.2.2.

5.2.1 Representation of the Environment

We have used a hybrid Quadtree-Waypoint approach for the representation of the environment for path-finding. A Quadtree is used for two reasons. Firstly, our agents are moving on a 2D plane and as a result an area based represen-

tation will be sufficient to represent the environment for path-finding. Secondly, a Quadtree representation is able to offer some run-time performance improvement to the path-planner because a Quadtree basically organise the environment in a hierarchical manner. Waypoints are used to represent a rough path for the agent to follow in order to reach the destination.

We will now outline the construction of our hybrid Quadtree-Waypoint representation of the environment for path-finding. In particular, we will first explain how a typical Quadtree can be automatically constructed. This is followed by an explanation of how our hybrid Quadtree-Waypoint representation can be used to improve the quality of the path generated by a path-finding algorithm.

The construction of a Quadtree is relatively easy. First, a bounding box capable of bounding the entire scene is calculated. This bounding box will then serve as the bounding volume for the root node of the Quadtree. The area represented by the root node is then divided into four equally sized regions. Each of these regions is then attached as a child to the root node with a bounding volume describing the divided region. If an obstacle is detected within the space bounded by the child node, then it will be further subdivided. The division is stopped when the contents of the region is homogeneous, i.e., contains all accessible space or inaccessible space. The division can also be stopped when the maximum number of tree level is reached. The maximum depth of the tree will then determine the computational overhead required for the path-finder to access the leaves of the Quadtree. Figure 5.5 illustrates the construction of a typical Quadtree.

The Quadtree therefore divides and groups the regions according to their

Figure 5.5: Construction of a Quadtree.

accessibility, i.e., whether the region are blocked by structural obstacles. The next step will be to determine the type of connectivity between the leaf nodes, i.e., which regions are considered to be neighbours from the agent's current position. Figure 5.6 shows two possible types of connection between the leafs of a Quadtree. The connection type determines the computational cost of search for a route through the environment and the quality of the generated path. For example, having eight connections will mean that the algorithm has to evaluate all eight nodes to determine the best branch to search for a path. At the same time, having eight connections will mean that the agent will turn more smoothly around sharp bends and appear more natural. To further illustrate this point, refer to Figure 5.7.

As it can be observed from Figure 5.7, an eight way connection generates a smoother path than a four way connection especially around sharper bends. However both representations have a possible intersection between the agent's 3D model with the structure's 3D model when the agent navi-

Figure 5.6: Basic connectivity.

Figure 5.7: Trace of route generated.

Figure 5.8: Proposed waypoint layout.

gates too closely with the walls when turning. This problem occurs because the connections are placed too near the walls itself without taking into consideration the body of the agent.

We therefore propose a *half-space waypoint overlay method* to overlay a mesh of waypoints onto the environment. Figure 5.8 shows the layout of the waypoints on a single leaf node of the Quadtree. The waypoints on the edge of the area represented by the leaf node are connected together using a 4 way connection method. Let a *leaf edge* represents any edge of the bounding box represented by a leaf node, then there are two exceptions to the connectivity rule. Firstly, if the leaf edge is shared with an inaccessible leaf node, then the waypoint from that leaf edge is removed from the mesh. Secondly, if more than two leaf consecutive edge that is shared by inaccessible leaf nodes then the waypoint at the center of bounding box is also removed from the mesh. The first exception is to create a clearance space between the agent's position and the obstacle. This clearance space will help avoid any potential intersection between an agent's 3D model with a structure's 3D model. The second exception is to allow for automatic path smoothing when navigating around corners.

Figure 5.9: Sample connection between regions.

Figure 5.10: Navigation around corners.

Figure 5.9 shows a sample connection between the different regions based on the two exceptions to the connectivity rule. Note that waypoints that are next to structural obstacles are removed from the waypoint mesh. Figure 5.10 trails the plotted path using the half-space waypoint overlay method to automatically smooth the agent’s path around bends. Note that the generated path enables the agent’s 3D model to maintain a reasonable distance from the structure’s 3D model.

We have therefore proposed a Quadtree-Waypoint hybrid representation of the environment for path-finding that will allow the path-finding algorithm to plot a path free from structural obstacles. This means that the agents can focus on avoiding other agents when navigating through the environment. The representation is also able to generate a smoother path than it would otherwise be possible without the half-space waypoint overlays on a Quadtree node. In the next section, we will discuss the heuristics used for evaluating a way-point as the candidate route to a destination.

5.2.2 Heuristics for the Path-finder

In the previous section, we have described our approach to representing the environment for path-finding. In this section, we will describe the heuristics used by our A* path-planner to generate realistic pedestrian behaviours.

We have used two heuristics in our path-planner, distance and crowd density. We can easily add more heuristics into our planner should the need arise. This property is important because humans are complex social creatures and will react differently under different circumstances. Hence it can be a herculean effort to model a generic pedestrian for all situations. Therefore,

it is important for the underlying path-finding algorithm to be customizable to better target the scenario required by a simulation.

The distance heuristic attempts to estimate the distance between the agent's current position and its destination. This heuristic is used to reflect the fact that pedestrians will in most cases attempt to move to their destination using the shortest path that they know of. According to Figure 5.8, the agent is able to move diagonally across the grid and hence the *diagonal distance* will be a reasonable choice for our distance heuristics. Let d_p be the diagonal distance from the current point to some destination point p , then the diagonal distance is given by

$$d_p = \max(|x - p_x|, |y - p_y|) \quad (5.1)$$

where x and y is the position of the waypoint nearest to the agent. The diagonal distance is a faster heuristic than the more commonly used Euclidean distance estimate because we do not need to perform the relatively expensive square root operation to find the estimated distance. Diagonal distance also *underestimates* the cost of travelling between two points, e.g., if the cost of travelling one along the principal axis is D then the cost of travelling diagonally on a square grid should be $\sqrt{2} * D$. This means the planner may attempt to explore diagonal routes first before routes along the horizontal and vertical axis.

The crowd density heuristic measures the relative density of agents within an area. This heuristic is used to reflect the fact that pedestrians have the tendency to avoid congestions when navigating through the envi-

ronment. This is because navigation through a dense crowd is usually more difficult and slower than navigation through a sparse population.

$$h(d_p, \rho_p) = d_p * w_{dp} + \rho_p * w_{pp} \quad (5.2)$$

The heuristics for our path-planner can therefore be described using Equation 5.2 where w_{dp} and w_{pp} is the weight of the distance and density heuristics respectively. ρ_p is the density of the agent through which the agent has to pass through to get to point p . When the agent encounters a congestion, there is a chance that the agent will want to replan its path in order to find an alternative route around the congestion. This reflects the fact the pedestrians may take a longer but less congested route in order to reach their destination faster. In general, the higher the value of value of w_{pp} , the higher the chance is for the agent to replan its path.

We have therefore outlined the heuristics used for the path-planning process and will further demonstrate the results of using our proposed macro-level navigation model in Section 7.6. In the next section, we will address how we can make minor modifications to the macro-level navigation model so as to model individual perception of the environment.

5.3 Path-Finding and Visual Memory

Each pedestrian in the real-world is expected to have their own abstract map of the environment. Most pedestrians are not expected to have a complete map of the environment in their mind, instead they will frequently have only enough information for them to navigate through the area. For example, a

pedestrian may have been using the same route to get to work for the past year and may continue to use that same route indefinitely until a new route is found. In fact the mind will unconsciously lead the person along a route the person is most familiar with. Most pedestrians are also not expected to always explore new routes to get to their destinations unless their old route has proven to be infeasible.

Therefore to model this pedestrian behaviour, we need to model the fact that each individual pedestrian has their own visual memory about the environment, i.e., each agent will need to have an abstract map that describes the environment. In particular, each individual within the pedestrian crowd is expected to have their own partial set of the abstract map used to represent the environment. Therefore for a realistic pedestrian simulation, it is necessary for the macro-level navigation module to tap information from this personal visual memory of each agent rather than to use information from a global representation of the environment.

Using the representation of the environment for path-finding in Section 5.2.1, the visual memory of the agent can be represented as a subset of the waypoint mesh. The path-planner will hence compute the cheapest route based on what the agent knows about the environment rather than what is available in the environment. This means that an agent may be found to be travelling along a local optimum path because the global optimum path is not known to the agent.

Figure 5.11 shows a simple environment where visual memory has been modeled for the agent. The patterned cells in the figure represent regions that the agent is ignorant about, i.e., the agent does not know that it is able

Figure 5.11: Visual memory and route choices.

to travel through the region to reach the destination. Based on the current knowledge of the agent (clear shaded cells), a longer route was traced from the agent's start point to the destination because the agent was not aware of the shorter (dashed) route. This reflects realistic human navigation because we do not always know the shortest possible path to the destination. Similarly, even if we do know the shortest path, we may just prefer to take the *usual* route because of old habits. Therefore the use of visual memory in the macro level navigation will allow for more realistic individual path finding.

However several challenges exist in the modelling of visual memory for individual agents. Foremost is the difficulty in constructing visual memory of every agent in the scene. In order to accomplish this, an automated approach of generating visual memory for each agent is needed. Secondly, representing the visual memory of individual agents can significantly increase the memory requirements of the simulation. Each agent with visual memory will for example need to have an instance of the waypoint mesh used to represent the environment. Memory requirements can be reduced by keeping an indexed copy of the waypoint mesh rather than the complete data structure of the waypoint mesh. Even so, the memory requirements for representing visual memory a scene with many agents can still be significant.

5.4 Summary

In this section, we have discussed the need for a macro-level navigation module to enable our agents to plan a path for our autonomous agents. We started our discussion with the different means of preparing the environment

for path-finding. We have also weighted the benefits and setbacks of using the different representations. Next, we have discussed about the three main planners used for autonomous agents both in games and other real-time simulations. Specifically, we have looked at several variations of the A* algorithm and discussed the use for each variation.

We have described our proposed macro-level navigation model that attempts to model the decision making process of a pedestrian during path-planning. Specifically, we outlined the use of a Quadtree-Waypoint hybrid representation of the environment for path-finding to represent a pedestrian's abstract representation of the environment. Two heuristics were then used to estimate the cost of moving between any two points on the scene; the distance heuristics is used to mimic a pedestrian's tendency to use a shorter path to reach the destination, the crowd congestion heuristic is used to reflect a pedestrian's tendency to avoid dense crowds when an alternative route to the destination exists. Finally, we have discussed the use of visual memory in our macro level navigation model and explained how visual memory can be used to increase the realism of our pedestrian behaviours.

We have thus discussed both the micro and macro level navigation module for our pedestrian. In the next section, we shall examine how our proposed navigation model for the agents can be used to model and simulate a dense pedestrian crowd.

6 Modelling and Simulation of Dense Pedestrian Crowd

Dense crowd simulation (Figure 6.1) is the simulation of a large number of agents that is concentrated within a small span of space. Dense crowd simulation does not necessarily mean that there are more agents in the environment but only more agents per unit space. In the simulation of a dense pedestrian crowd, we are interested in the simulation's ability to

1. Demonstrate realistic navigational behaviours similar to pedestrians in a dense crowd;
2. Balance computational load in order to maintain the frame rate of the simulation;
3. Balance the memory requirements necessary to run the simulation on a single workstation.

However, a few differences exist between a dense crowd simulation and the pedestrian simulation that we have discussed so far. In particular, the agents will have a harder time to navigate through a dense crowd due to the higher chance of being on a collision course with another agent in the scene. This means that the agent will need to make steering corrections more frequently, e.g., taking more twists and turns to reach the destination. As a result, the run-time performance of a dense crowd simulation may be inferior to a normal pedestrian simulation due to the much higher frequency of inter-agent collision avoidance.

Figure 6.1: Peak hours crowd at Penn Station, New York.

The modelling of realistic pedestrian behaviour in a dense crowd is also the first step towards modelling a realistic crisis evacuation scenario. For example, the ability of the pedestrian simulation to reproduce crowd level behaviours such as *herding* will directly affect the believability and accuracy of the simulation. The accuracy of the simulation will then in turn affect simulation results such as total time to evacuation and casualties from the evacuation. The applications for a dense crowd simulation is not restricted only to crisis simulation. For example, dense crowd simulations also can be used to study the adequacy of building exits by simulating pedestrian flows through different parts of the building during peak hours (Figure 6.1).

In Section 6.1, we will first discuss the limitations of our micro-level navigation module in generating the navigation behaviours for our pedestrians in a dense crowd. We will then explain how the micro-level navigation module can be extended to resolve the problem of avoiding multiple agents concurrently. In Section 6.2, we will discuss the various behaviours that can be observed from a dense pedestrian crowd and how behaviours like lane

formation can be generated by our micro-level navigation module. We will then conclude our discussion on the simulation of dense pedestrian crowd in Section 6.3.

6.1 Extending the Micro-Level Navigation Module

In a dense pedestrian crowd, there will be an increase in collision avoidance between multiple pedestrians, i.e., the agents will need to avoid more than one agent concurrently when navigating through the crowd. However, our existing micro-level navigation module is not capable of considering the movement of multiple agents together. As a result, we need to extend our previously proposed micro-level navigation module in order to account for the speed and orientation of all the neighboring agents around the source agent. In Section 6.1.1, we will first explain with greater detail the limitations of our previously proposed micro-level navigation module. Then in Section 6.1.2, we will outline the extension necessary for the source agent to consider the movement of multiple agents when navigating through a dense pedestrian crowd.

6.1.1 Limitations of the Micro-Level Navigation Module

Navigation through a dense crowd can be a tiring experience as pedestrians need to be on a constant alert to avoid collision with neighbouring pedestrians. As a result, movement through a dense crowd is generally slow and lane-forming. In certain cultures for example, social norms may form voluntary lanes for flow efficiency such as the scenario shown in Figure 6.2. In

Figure 6.2: Voluntary lane formation in Shinjuku Train Station, Tokyo.

other cultures where such social norms are absent, then ad-hoc lanes may be formed, [107]. Hence lane formation is a naturally occurring crowd phenomena because it is easier to move in the direction of the traffic flow than to move against the flow.

In our two-tiered navigation model for our agents, navigation through a dense crowd is achieved by the micro-level navigation module. The current micro-level navigation module enables the agent to avoid collision by considering only the target agent with the highest collision threat. For a sparse crowd, this approach works well because there is sufficient time for the agent to avoid collision with other agents in the environment. Furthermore in a sparse crowd, the probability of consecutive collision avoidance is relatively slim. Therefore, a single target collision avoidance constraint is sufficient for an agent to avoid the collision in a sparse crowd.

However for a dense pedestrian crowd, multi-agent collision avoidance is necessary to prevent consecutive steering corrections which can be com-

putationally expensive and is generally unrealistic. Furthermore consecutive steering corrections can cause the agent to oscillate collision avoidance between neighbouring agents as illustrated by Figure 6.3. $\vec{v}_{ri}$ is the relative velocity between the source and target agent i and $\vec{v}_{gi}$ is the velocity of target agent i . The source agent is initially found to be on a collision course with target agent 1. As a result, the source agent determines that there is a need to perform avoidance maneuvers so as to avoid target agent 1. However, the micro-level navigation module will compute the new steering parameters for the source agent without considering the headings of target agents 2 and 3. As a result, the newly computed velocity of the source agent in fact leads to a collision course with target agents 2 and 3 (Figure 6.3(b)). This means that the source agent will again need to make steering corrections during the next update cycle. Using the same argument, the source agent may eventually end up in a deadlock with all the target agents, i.e., the source agent may oscillate between avoiding target agents 1, 2 and 3 without being able to actually avoid all of them.

One method of resolving this deadlock is to search for a value of $\theta_r(t + \Delta t)$ that will not cause a collision with all other surrounding agents. However without a proper guiding heuristics, this brute force approach will be computationally expensive. Hence for the purpose of computational efficiency, the micro-level navigation module should be a one-pass steering algorithm, i.e., the algorithm should be able to analyse all the constraints during one iteration and compute the new steering parameters for the agent during the same iteration.

Therefore, for the agent to be able to avoid all the surrounding agents

Figure 6.3: Three relative frames between different target agents.

that it is aware and to prevent deadlock, we need to develop navigation constraints on the source agent for *each* of the surrounding agents, i.e., one such constraint is needed for each target agent that the source agent is aware of. These constraints should be the terms of ω_s , so that it can be combined with the other constraints that we have discussed thus far. The resultant steering parameters would then have considered the relative position and velocity of all the target agents around the source agent in its computation. The new steering parameters will hence result in a path that is free of collision from all the observed target agents.

6.1.2 Extended Collision Avoidance Constraint

In Section 4.1.2, we have outlined our approach for avoiding one target agent. However, when modelling a dense pedestrian crowd, it may be necessary for the source agent to avoid more than one target agent at any one time. This is because, the source agent's newly computed steering parameters may be

able to successfully avoid a particular target agent but be on a collision course with another nearby target agent. We are only concerned with target agents that the source agent is aware of, meaning that collision is actually possible between agents who are not aware of each other even when they are near each other. This behaviour will generate a more realistic simulation because it naturally reflects how a real pedestrian actually moves around the environment.

In our proposed model, changes in a relative frame between a source and target agent can affect other relative frames between the source agent and other target agents. As a result, the first step to achieving a multi-agent avoidance constraint is to analyse the effects of changes in one relative frame on one other relative frame of the source agent. Let the relative frame between the source agent, a_s , and the i^{th} target agent a_{gi} be R_{sgi} . Given two target agents, a_{g1} and a_{g2} , then the effect of avoiding a_{g1} on the relative frame between the source and target agent a_{g2} can be described by

$$s_{r1} = \frac{s_{s2} \sin(\theta_{g2} - \theta_{r2}) - s_{g1} \sin(\theta_{g1} - \theta_{r2})}{\sin(\theta_{r1} - \theta_{r2})} \quad (6.1)$$

where, $s_{g2}(t)$ and $\theta_{g2}(t)$ are the speed and orientation of the second target agent. $\theta_{r2}(t)$ is the angle of the relative velocity between the source and second target agent. Interested readers may refer to Appendix 8.2 for the proof of this relationship.

Equation 6.1 can be used to describe how the changes in the relative speed, s_{r1} , from the relative frame, R_{sg1} , affects the angle of the relative velocity, $\theta_{r2}(t)$, from the relative frame, R_{sg2} . From Section 4.1.1, we can

observe that collision can be simultaneously avoided between a_{g1} and a_{g2} if we ensure that both $s_{r1}(t + \Delta t)$ and $s_{r2}(t + \Delta t)$ lies outside of their respective collision zones. To this end, we will need to ensure that $s_{r1}(t + \Delta t)$ does not result in a $\theta_{r2}(t + \Delta t)$ that falls within the collision zone of the source agent and the target agent a_{g2} .

From Section 4.1.1, we can observe that it takes two values of θ_{r2} to describe a collision zone between a_s and a_{g2} . Therefore, using Equation 6.1, we will obtain two corresponding values of $s_{r1}(t + \Delta t)$ that describes the collision zone between the source and target agent a_{g2} . Each value of $s_{r1}(t + \Delta t)$ will then have a corresponding $\theta_s(t + \Delta t)$, where $\theta_s(t + \Delta t)$ can be found using Equation 6.2

$$\tan \theta_s = \frac{s_{g1} \sin \theta_{g1} + s_{r1} \sin \theta_{r1}}{s_{g1} \cos \theta_{g1} + s_{r1} \cos \theta_{r1}} \quad (6.2)$$

Let the two values of $\theta_s(t + \Delta t)$ be θ_{m1} and θ_{m2} such that $\theta_{m1} \leq \theta_{m2}$. Then the second target agent can be avoided by adding constraint, l_4 , as defined by

$$l_4 = \neg(\theta_{m1} - \theta_s \leq \omega_s < \theta_{m2} - \theta_s) \quad (6.3)$$

Therefore to achieve multi-agent collision avoidance, each target agent that the source agent is aware of will require a l_4 constraint. For example, a source agent with five visible target agents will require one set of avoidance constraint and four sets of l_4 constraints to avoid all target agents.

6.2 Pedestrian Behaviours in a Dense Crowd

In a dense pedestrian crowd, the necessity of the situation has resulted in several unique behaviours that are visible only in a such a situation. In particular, we will describe how voluntary and ad-hoc lane formation behaviours can be naturally realised using our proposed micro-level navigation model in Section 6.2.1.

We will then discuss the queuing (Section 6.2.2), pushing (Section 6.2.3) and falling behaviours (Section 6.2.4) that are commonly seen in a dense pedestrian crowd. However these behaviours require more specialized rules that cannot be achieved by simply adding new constraints on our proposed micro-level navigation model. While queuing, pushing and falling behaviours are not exactly part of a pedestrian’s navigation behaviour, they are briefly discussed for their importance in achieving realistic dense pedestrian simulation. We will also explain why these behaviours serve as interesting future extensions of our proposed pedestrian navigation model.

6.2.1 Lane Formation Behaviour

Lane formation behaviour is a naturally occurring phenomenon among pedestrians in a crowded situation. In particular, lanes are formed by pedestrians to ease their own movements through the environment. Figure 6.4 explains how lanes can be formed using our proposed micro-level navigation model. Note that lanes are formed due to the actions of different agents rather than due to an explicit rule. As two groups of pedestrians from opposite direction approaches, some agents may attempt to navigate pass the other group of

pedestrians by navigating to the left or right of the group as a result of the basic collision avoidance behaviour. Other agents then begin to either follow or overtake the avoiding agent ($t = 20$), whose collective behaviour finally resulted in lanes being formed ($t = 30$).

Lanes can be formed in an ad-hoc manner or they can be formed in a voluntary manner. Ad-hoc lane formations are lanes formed when pedestrian flows from two or more different directions meet. Typically an ad-hoc lane will only persist for as long as the following agents decide to use the lane. Ad-hoc lanes are characterised by short-lived and irregular lanes in a dense pedestrian crowd. In particular, ad-hoc lanes can be commonly seen in crowded trade fairs and exhibitions such as Figure 6.5.

Voluntary lanes are formed when pedestrians keep to their own side of the pedestrian facility depending on their direction of movement. Voluntary lanes are characterised by more defined and persistent lanes in a dense crowd. Therefore unlike ad-hoc lane formation, voluntary lane formation occurs when all agents in the environment volunteer to keep to their own lanes as they move through the environment. As a result, the lanes formed are relatively long lived.

In fact, a pedestrian may even be frowned upon if they start obstructing pedestrian flows by not following these voluntary lanes. As a result, pedestrians may form a habit on how to move through a particular environment, such as keeping left (or right). This habit when reinforced with other experiences of the pedestrian's daily life creates a kind of *social norm* that helped establish these voluntary lanes. Voluntary lanes are frequently found in commuter areas (Figure 6.2) such as train stations and crowded

Figure 6.4: Lane formation behaviour.

Figure 6.5: Ad-hoc lane formation during the Tokyo Game Show.

underground passes.

6.2.2 Queueing Behaviour

Queue behaviour can be observed in real-life situations such as exiting a tightly packed room and queuing for train tickets (Figure 6.6). Queueing behaviour requires the agents to

1. Identify the need to start queuing, such that it is no longer possible for the agent to move forward;
2. Select the best lane to queue in order to complete the agent's objective in the shortest possible time;
3. Recover from the queuing behaviour and move forward as much as possible until there is a need to stop and wait again.

Impatient agents are also expected to make use of queue cutting movements so that they can reach the goal faster. Concerned agents may also make use

Figure 6.6: Ticket queuing at Suzhou Railway Station, China

of movements that prevents queue cutting so as to reduce their own waiting time. Queuing therefore pose an interesting problem to the modelling of pedestrian behaviours because the agent now need to consider more factors when deciding on where to move next.

6.2.3 Pushing Behaviour

The pushing behaviour is an act of pushing through the people in front in order to reach a destination. Unlike the collision avoidance behaviour, pedestrians that are pushing will not observe the personal space of those around them. For example, when the movement through the area is restricted by a narrow exit or entrance, some pedestrians will attempt to *push* through the crowds in front so as to reach the exit first. In reacting to the pushing, some pedestrians may allow the pushing pedestrian through by stepping aside or by pushing their own neighbours aside. On the other hand, some pedestrians

Figure 6.7: Pedestrians pushing to board a train.

may attempt to block the advances of the offending pedestrian by offering resistance. The advance of the pushing pedestrian may therefore be halted if the pedestrian is not able to overcome the resistance offered by other pedestrians. In order to model the pushing behaviour, our agents will therefore need to consider how the external forces from the pushing agents affect their own movements, e.g., whether they will offer a resistance to the push or allow the pushing agents through. Figure 6.7 shows the pushing behaviours among pedestrians trying to board a crowded train.

6.2.4 Falling and Trampling Behaviour

As a result of being pushed by other pedestrians in a dense crowd, weaker pedestrians may fall down when they are overwhelmed by the crowd. Pedestrians who have fell down behave like an immobile obstacle that may slow down the movements of other pedestrians. During a normal situation, some

pedestrians may attempt to help the fallen pedestrian up, although this may not always be true in panic situations. For example, those who fell down in a panicked crowd, risk being trampled by other agents who are trying to step over the fallen agent to reach their destination [44]. Therefore falling and trampling behaviours are important pedestrian behaviours required in the realistic simulation of a pedestrian crowd under both normal and panic situations. Similar to the queueing behaviour, the agent will need to consider the effects of external forces on its own movements in order to demonstrate the falling behaviour.

6.3 Summary

In this section, we have looked at the modelling and simulation of a dense pedestrian crowd. We have discussed about the differences between the navigation of a pedestrian through a sparse crowd and through a dense crowd. Using the differences, we raised the limitation of our previously proposed micro-level navigation model. As a result, we derived a new constraint on the orientation of the agent so that pair-wise collision avoidance between all the neighbouring agents and the source agent can be achieved. However, having a new constraint will reduce the performance of the simulation due to the additional computations required for the constraint. Therefore, while this constraint will produce more *accurate* pedestrian navigation behaviours, it can sometimes be ignored if performance is a consideration. We will future illustrate this concept in Section 7.9.3, where we will examine the effect of agent's missing an avoidance with another agent on the results of the simula-

tion. We have also examined some of the behaviours common for pedestrians in a dense crowd. Specifically, we explained how lanes can be formed in a dense pedestrian crowd using our agent's micro-level navigation module.

7 Implementation and Experimental Results

We have discussed our agent’s representation and explained the importance of a realistic navigation model in order to generate realistic navigation behaviours for our agents. In this section, we shall combine those ideas together and demonstrate our agent’s navigation capabilities using a 3D pedestrian simulation.

We will first describe some memory management considerations that was used to improve the performance of our simulation in Section 7.1. We will then describe in Section 7.2, the testing scenarios that we have created for our pedestrian simulation. These scenarios have been selected because they represent different real-life environments and provide different types of challenges. Section 7.3 shows the basic navigation behaviours of our agents using only the micro-level navigation model. Section 7.4 demonstrates how individual personality can be reflected using our micro-level navigation model by adding constraints that reflect the mental states of a real pedestrian.

We will then present the results of a dense crowd simulation in Section 7.5. In particular, we show how *ad-hoc* and *voluntary* lanes can be formed by our agents. In Section 7.6, we will combine the macro-level and micro-level navigation model and demonstrate our agent’s ability to avoid both structural obstacles and other agents. We have also implemented a shopping mall scenario where different agents have been pre-assigned with different roles.

The computational performance of our pedestrian simulation will be dis-

cussed in Section 7.7. Then we will continue to discuss the two performance optimisation techniques that we have implemented, free-lists (Section 7.8) and time-slicing (Section 7.9). Finally, we will close our discussion in Section 7.10.

7.1 Memory Management

Memory management is primarily concerned with the efficient allocation and deallocation of different types of memory. In particular, we are concerned with the memory management techniques for managing the system's RAM. Memory management is important to real-time pedestrian crowd simulation because

1. Crowd simulation requires many dynamic memory allocations and deallocation which can significantly degrade the performance of the simulation;
2. Crowd simulation requires many resources to be stored on the system's RAM which can easily cause the system to run out of memory;
3. Crowd simulation contains many instances of the same object and efficient memory management of these objects can improve the performance of the simulation.

Therefore, due to the importance of RAM to the operation of the entire computing system, it is common for operating systems to have some form of built-in memory manager that takes care of interfacing between the programs and the actual memory hardware. Our discussion for memory management

is therefore restricted to the services provided by the operating system for memory allocation rather than the actual communication with the memory hardware.

The key objective of a memory manager is to ensure that sufficient memory space is available to the simulation for successful execution. However generic memory managers provided by most operating systems do not know the exact memory requirements of the process until the process actually requests for them. At the same time, memory is organized as memory blocks of fixed size and memory managers are not capable of allocating memory less than one block of memory. Although the underlying operating system hides this fact from a programmer, the problem of *memory fragmentation* as a result of allocating memories larger than the required memory size still exists. We will continue to discuss more about memory fragmentation in Section 7.1.2.

Our research into efficient memory management is primarily motivated by the fact that there is high number of memory related operations in a real-time crowd simulation. For example in our path-finding algorithm, open and close lists are used to track search spaces on the path-finding graph. These lists expand and contract dynamically, and as a result require many memory allocations and deallocation to complete the planning process. However memory management is a large field of research by itself, hence we will not be able to cover all existing technologies in this section. Interested readers may refer to Wilson's excellent survey on memory management for a more thorough review [124].

We will first discuss the different methods of allocating memory in Section 7.1.1. Then we will explain the concept of memory wastage and memory fragmentation in Section 7.1.2. Finally, we will explain the importance of the memory manager's search operations on the overall performance of a pedestrian crowd simulation.

7.1.1 Allocation Strategies

Allocation strategies are concerned with the method of selecting free memory blocks to fulfill the memory requests. Memory allocation can be broadly classified into three categories [53]. Sequential fit based approach attempts to search the memory space for a memory slot that is capable of fulfilling the memory request. There are two basic strategies for sequential fit algorithms. First fit approaches will return to the application the first memory space encountered by the manager that is large enough to satisfy the memory requirement. On the other hand, the best fit approaches attempt to find the best memory size for the request to ensure that there is minimum memory wastage. Several existing improvements of these two basic strategies exist. For example, Knuth suggested a performance improvement over the first fit algorithm by starting the search from the address location of a previously successful search [60]. In this way Knuth attempts to reduce the search duration necessary to find a suitable memory location to fulfill the memory request. However according to Johnstone Knuth's an approach actually increases memory wastage [53] (see Section 7.1.2).

The second memory allocation strategy uses *free-lists*. *Free-lists* at-

tempts to reduce the search duration for a free memory space by grouping free memory blocks into separate lists. Each list will contain all the free memory blocks within a range of sizes. For example if a 75 kilobyte (kb) memory block has been deallocated by a program, then the memory block will be placed on a *free-list* containing free memory blocks between 50-100kb in size. As a result, *free-lists* are best suited for managing memory where all the objects have the same size. Depending on the specification of the *free-list* strategy, several free blocks from a smaller sized list can be aggregated together to fulfil a request for a larger memory block.

The third type of memory allocation strategies is called the *Buddy system* which originates from the work of Knowlton [59]. The buddy system uses a hierarchical approach to grouping the memory space which is similar in concept to the Quadtree used in our macro-level navigation module. The memory space is iteratively divided until the lowest memory unit is reached. At each level of the memory tree, nodes are paired up to form *buddies*. Such pairing helps to ensure that the memory space returned to the process consist of a contiguous block of memory. Memory requests are fulfilled by grouping pairs of *buddies* together until the combined memory space is large enough to satisfy the request. Unlike the *free-list* strategy, the *buddy system* only allows groupings that are logically (based on memory address) adjacent to each other.

7.1.2 Memory Wastage

System memory is a limited resource and one of the roles a memory manager is to minimise memory wastage. Memory is organized as units of fixed sizes this means that whenever a memory request is made, the memory manager will allocate memory that is above the requested memory resulting in memory wastage. Since the actual required memory is less than the allocated memory, the result of having unused memory within a memory block is called *internal memory fragmentation*, i.e., memory wasted within a memory block. Internal memory fragmentation can be a concern for crowd simulation due to the large number of instances of the similar objects with the exact memory requirement. For example, if each instance of the object wastes 10kb of memory due to internal memory fragmentation, then one hundred instance of the same object will waste 1000kb of memory. As it is not possible to allocate half a memory block to the application, internal memory fragmentation can be better reduced through software design changes (e.g., using an array) than through memory allocation strategies.

External memory fragmentation occurs when a free contiguous block of memory gets divided into many smaller fragments. This results in many small free memory blocks to exist between these used memory blocks. It is therefore possible that a process may never require such a small fragment of memory. As a result, having these small fragments of memory is as good as having none. External memory fragmentation can be reduced through the use of correct memory allocation strategies. For example according to Johnstone, a buddy system memory allocation strategy can cause an average of

50% of the memory to become unusable, while a best fit memory allocation strategy causes only an average of 2% of the memory to be lost due external memory fragmentation [53]. External memory fragmentation can cause the memory allocation process to slow down significantly because the memory manager will need to search more empty spaces before a suitably sized memory can be found. Therefore in a real-time crowd simulation, external memory fragmentation can cause the simulation's performance to degrade after some time.

7.1.3 Search Performance

Memory allocators basically search the memory space for free memory blocks to satisfy the memory request of a process. Therefore two factors affect the search efficiency of the memory allocator, the organisation free memory space and the search algorithm used to find the free memory space. Free memory space can be organised using a variety of methods ranging from a simple linked-list of memory address to specialized data structures such as the randomized binary search trees [105]. The purpose of memory organisation is to reduce the time needed for the memory manager to locate a suitably sized memory slot to fulfill the memory request.

Therefore a best fit memory allocation strategy together with a link-list organisation of the free memory space is equivalent to a brute-force sequential search for free memory slots. On the other hand, using the best fit memory allocation strategy with a hash-table organisation of free memory space may produce a better run-time performance. Since a crowd simulation will require

many dynamic allocations of memory, the search performance of the memory manager has a direct impact on the performance of the crowd simulation. Hence we require the computational overhead due to the memory allocation process should be kept to a minimum in order to boost the performance of a real-time simulation.

7.1.4 Memory Management for Crowd Simulation

We have stated that a crowd simulation has many operations that require dynamic allocation of memory. Hence to improve the performance of our simulation, we have determined that it is necessary to have some form of memory management. Since processes such as path-finding operates on the system's RAM, our memory management strategy will be focused on efficient memory allocation from the system's RAM. At the same time, the memory requests for these processes are for data structures that have exactly the same memory size. Hence the selected strategy should be able to utilise this special property to minimise the time needed to search for a suitable memory slot. Examples of data structures with exactly the same memory size in a crowd simulation will include the vertices of the agent's 3D model and the nodes used for the agent's path-finding.

As a result, we propose that *free-list* is the most suitable strategy for dynamic memory allocation for our crowd simulation. This is because *free-list* is designed for managing memory request where the memory requirements of all the objects are exactly the same size. *Free-lists* are also easy to implement and is hence a good memory allocation strategy in situations where only a portion of the application require memory management.


```

Function void* Allocate(){
    if(nFreeListTop < 0)
        ExpandFreeList();

    return pFreeStack[nFreeListTop--];
}

Function void DeAllocate(void* ptr){
    pFreeStack[++nFreeStack] = ptr;
}

```

Figure 7.1: Pseudocode for memory management using *free-list*.

```

Function void ExpandFreeList(){
    //ELEMENT_SIZE = size of each record
    //BLOCK_SIZE = number of records to allocate
    //Creates a Linked-list of memory blocks
    pMemoryBlock = new char[ELEMENT_SIZE*BLOCK_SIZE];
    LinkedList->add(pMemoryBlock);

    nTotalCapacity += BLOCK_SIZE;

    //Add Newly created memory blocks to the freelist stack.
    for(int i = 0; i < nBlockSize; i++){
        pFreeStack->Push(&newNode->pMemoryBlock[i]);
    }
}

```

Figure 7.2: Pseudocode for the expanding the *free-list*.

There are two possible implementations for *free-lists*, stack-based first-in-first-out or queue-based first-in-last-out. We have used a stack-based approach to implement our *free-lists*. Figure 7.1 shows the pseudocode for the allocation and deallocation of memory for a stack-based implementation of *free-lists*. Note that searching is not required to find a suitable sized memory space during the allocation process. Instead the top-most free memory address on the memory stack is returned. This is the primary computation advantage of implementing a *free-list* based memory manager.

Usually a *free-list* will reserve a contiguous block of memory from the main memory and keep it until the program terminates. If the application requires more memory than the *free-list* has reserved, then there are two methods of obtaining more memory. The first method will be to relocate the existing reserved memory to a new and larger block of memory. The second approach is to reserve another contiguous block of memory and track these blocks using a linked-list. External memory fragmentation can be reduced if a large contiguous block has been used, but memory relocations can be an expensive task. Furthermore, all memory pointers that are pointing to the old memory address must be updated to the new address to prevent access of invalid memory locations. We have used the linked list approach to take advantage of faster memory expansion for the free-list. Figure 7.2 shows the pseudocode used for reserving additional chunks of memory using linked lists. In Section 7.8, we will demonstrate the significance of having a memory manager for a pedestrian crowd simulation.

7.2 Simulation Scenarios

Different scenarios have been designed to represent the different real-life environments where pedestrians can commonly be found. In particular, wide open spaces (Section 7.2.1), narrow spaces (Section 7.2.2), indoor areas (Section 7.2.3), and urban areas (Section 7.2.4) have been implemented. We have selected these scenarios so that we will be able to demonstrate the different navigation behaviours of pedestrians in various environments. In this section, we will describe the applications for each of these environments and present the challenges posed by these environments towards the modelling of a realistic pedestrian crowd.

For all experiments, we have used the Game Studios 6 engine [1] for rendering the agents and the environment. The simulation was bench-marked on an Intel Centrino M 1.6GHz computer with 1.25GB RAM with an integrated graphics card. The agents behaviours are programmed using C++ and compiled with Microsoft Visual Studio Express 2005.

7.2.1 Wide Open Spaces

Figure 7.3 shows a real-life open area in the city of Beijing. Pedestrian navigation in such an area is unique because there is so much space to move around, unconstrained by structural obstacles. Pedestrians in the environment will only need to avoid other pedestrians to move through the environment. As a result, the ability to generate realistic navigation behaviours for pedestrians moving in wide open spaces is the first step towards generating realistic navigation behaviours for other environments.

Figure 7.3: Tian'an men Square in Beijing, China.

Figure 7.4: Pedestrian simulation in wide open spaces.

Figure 7.4 shows our implementation of pedestrian navigation in a wide open space. Note that this scenario represents an *unaligned* environment, i.e., agents can move in whatever direction they want without the need to follow any lanes. This is also the simplest environment possible for a pedestrian simulation.

Figure 7.5: Crowded street along a popular shopping district.

Figure 7.6: Pedestrian simulation in simple enclosed spaces.

7.2.2 Narrow Spaces

The second scenario attempts to model pedestrian movement in a restricted space such as a corridor or a street (Figure 7.5). Typical structural obstacles will be in the form of doorways and walls that constrain the movement of pedestrians in a particular direction. The environment is still relatively simple and most pedestrians may not need to avoid structural obstacles while navigating through such an environment. Simulation of pedestrian flows through a narrow space can be useful for planning evacuation routes.

Figure 7.7: Interior of a typical shopping mall.

Figure 7.6 shows our implementation of pedestrian navigation in a narrow space. The color assigned to the agents represents the general direction in which they are moving. The color assignment is arbitrary whose primary purpose is to allow users to trace the movement of pedestrians moving in a particular direction. Note that the shape of the environment causes the agents to move in a partially *aligned* environment, i.e., there is a general direction which the agents are arriving from and moving towards. As a result, narrow space pedestrian simulation is useful in studying directional pedestrian flows. In fact, most of our dense crowd simulations have been implemented using narrow spaces.

7.2.3 Indoor Areas

Indoor pedestrian simulation focuses on generating realistic navigation behaviours for pedestrians distributed within an indoor area such as train stations and shopping malls. Typical features within an indoor area may include

Figure 7.8: Pedestrian simulation in a shopping mall.

decorative structures, shops and rest areas. Pedestrians in the environment may try to visit some *attractions* such as shops or may simply be navigating through the area. There are several applications for pedestrian simulation in indoor areas. For example, simulation of the pedestrian flows can help the building management to increase the shopper's flow through different areas of the shopping mall. They can also use the information to help set the rent of tenants in different areas of the mall. It will also be possible to simulate the evacuation process of pedestrians in the mall and study the adequacy of various evacuation routes.

We have implemented a shopping mall for our pedestrian simulation as shown in Figure 7.8. Agents are required to avoid both structural obstacles and other moving agents together when navigating through the environment. In such an environment, the macro-level navigation model is used by our agents for path-planning and avoidance of structural obstacles. The micro-level navigation model has also been used by our agents for inter-agent collision avoidance and to steer the agent, with the aid of a rough plan outlined by the macro-level navigation model, towards its destination.

7.2.4 Urban Areas

Simulation of pedestrian flows in urban areas such as pedestrian zones (Figure 7.9) and central business districts have many applications in the area of building and crisis management. For example the simulation can be used to simulate pedestrian flows during festivals where the simulation will be useful in the planning of the layout for stalls in the festival. The information also allows planners to maximise pedestrian flow through the different parts of the festival. Pedestrians in these environments are assumed to either have destinations to a particular building in the environment or are just passing through the area. Typical structural obstacles in such an environment will include buildings and decorative objects such as fountains and statues.

We have implemented a simplified urban environment as shown in Figure 7.10, to experiment with pedestrian flows in urban environments. In particular, we tested the ability of our pedestrians to avoid odd shaped structural obstacles as they plan the way through the environment. Both the micro-level and macro-level navigation model are required for our agents to navigate through an urban environment.

Figure 7.9: A pedestrian zone in central Munich, Germany.

Figure 7.10: Pedestrian simulation in a urban area.

7.3 Basic Pedestrian Behaviours

Several stand-alone scenarios using a simple wide open environment (Section 7.2.1) have been implemented to show some of the navigation behaviours that our micro-level navigation model can demonstrate. The behaviours demonstrated in this section are *emergent behaviours* because the agents are not explicitly taught to recognise and react to a particular configuration. Instead, these behaviours emerge as a result of applying and combining the collision avoidance, orientation and speed constraints on an agent based on its *relative frame*.

Four of the more common pedestrian behaviours will be demonstrated in this section, they are the *avoidance*, *seperation*, *overtaking* and *speed matching* behaviours. These behaviours also form the basic building blocks of more complex navigation behaviours like *lane formation*. We will illustrate each of these behaviours using key frames captured from our pedestrian simulation and explain how they can be achieved by our proposed micro-level navigation model.

7.3.1 Simple Avoidance Behaviour

In this scenario, we have modelled two agents in a wide open area on a heads-on collision course. We will first demonstrate the sensors that are attached to the agent and how they are used in our pedestrian model. We will then show that there can be more than one way of avoiding a simple collision, i.e., the outcome of an *encounter* depends upon the actions taken by either of the colliding agents. Collision avoidance behaviour is achieved by our agents

Figure 7.11: AABB, personal space and vision samplers.

using a combination of collision avoidance and speed constraints described in Sections 4.1.2 and 4.1.3. In particular, the speed constraints represent the comfortable walking speed of an agent, e.g., $0 \text{ ms}^{-1} \leq s_s(t) \leq 0.8 \text{ ms}^{-1}$.

Figure 7.11 shows a trace of the agent's AABB, *circular personal space* and *ray-based vision samplers*. The AABB has been used to model the agent's touch sense; the agent will sense actual physical contact if another bounding volume intersects its own AABB. The AABB is also used by other agents to sense the presence of the agent in the environment. The AABB is used instead of the agent's actual geometry so as to improve the performance of the simulation by allowing for faster collision detection.

The *circular personal space* has been used to represent how much clearance the agent will put between itself and other agents in the environment. In a dense pedestrian crowd, the personal space will have a smaller radius, while in a sparser setting, the personal space will be larger. In real-life, pedestrians also will accept a smaller personal space between themselves and those they know, such as family and friends, as compared to strangers.

The *ray-based vision samplers* are rays of fixed vision strength that originates from the agent and limited in length by the agent's vision strength. These rays are tested with the scene graph to sense the presence of other

Figure 7.12: Collision avoidance behaviour.

objects in the environment. Figure 7.11 also demonstrates how each agent takes a side step to avoid the predicted collision. The left agent is able to predict the collision at $t = 10$ when the presence of the right agent was sensed (dotted lines) by the left agent.

Figure 7.12 shows that avoidance can also be performed by only one agent while the other agent continues its current headings unaffected. There are several ways of achieving this effect in our pedestrian model. For example, we can let one agent react to a collision more urgently than the other, or we can adjust the vision strength of both agents so that one of the agents will predict the collision before the other. In real-life, this is translated to the choices that pedestrians can make during their navigation, like how far away the avoiding maneuver should be made, and whether the avoidance should be made at all. The answers to these questions will depend upon the goals and personality of the pedestrian.

7.3.2 Seperation Behaviour

In the second scenario, we will demonstrate the seperation behavior when two or more agents are on a collision course with each other. Again the separation behaviour can be realised by using only the avoidance and speed

Figure 7.13: Separation behaviour.

Figure 7.14: Another way of avoiding the same collision.

constraints discussed in Sections 4.1.2 and 4.1.3. Specifically, each agent is again given a speed constraint that describes their comfortable walking speed and the ability to generate collision avoidance constraints to avoid collision with other agents.

Figure 7.13 shows the *seperation behaviour* as one of the natural behaviours that a pedestrian will take in order to avoid collision with approaching pedestrians. In addition, the *seperation behaviour* is also one of the basic building blocks that constitutes the *lane formation behaviour* in a dense pedestrian crowd which we have described in Section 6.2.1. Figure 7.14 shows a different pedestrian response for different runs of the same scenario. This result reflects the fact that pedestrian movement is an interactive process such that the outcome of an *encounter* between two or more pedestrians depends upon the actions that are taken by either pedestrian.

Figure 7.15: Speed Matching behaviour.

7.3.3 Speed Matching Behaviour

The *speed matching behaviour* is the act of matching the speed of a slower moving agent in front so as to avoid a collision. The *speed matching behaviour* is also one of the basic building blocks that constitutes the *lane formation behaviour*. In real-life, the decision to overtake or to follow may depend upon the situation and the pedestrian's personality. For example, a pedestrian who sees the traffic lights turning red will see no point in overtaking the pedestrians in front because the pedestrian will have to wait for the green lights anyway. Similarly, a pedestrian engrossed in a conversation with another pedestrian will not want to overtake the pedestrians in front because overtaking may disrupt their conversation.

Figure 7.15 trace the position of an agent who eventually chooses to match the speed of the front row of agents so as to avoid collision. The decision to overtake or follow depends upon the agent's minimum speed constraint for the situation. Specifically, if $s_{s\min} < s_g(t)$, where $s_g(t)$ is the speed of the slower moving agent, then an agent will be more willing to match the speed of the slower moving agent to avoid the collision. Furthermore, the front row of slower moving agents is not aware of the faster moving agent and as a result does not take steps to change their speed or movements to

Figure 7.16: Overtaking behaviour.

avoid the faster moving agent.

7.3.4 Overtaking Behaviour

The *overtaking behaviour* is the act of picking up speed so that a faster moving agent can move pass a slower moving agent. The *overtaking behaviour* is an important building block for the *lane cutting maneuver* found in dense crowds which we will describe in Section 6.2.1. Like the *matching speed behaviour*, the decision to overtake depends upon the situation as well as the agent's personality. For example, the agent may be running late for work and will need overtake slower moving pedestrians to be on time for work.

Figure 7.16, shows one such the execution of the *overtaking behaviour* by an impatient pedestrian. This behaviour can be achieved in our model by giving the agent a higher minimum speed constraint to denote its reluctance to slow down and following behind another agent. For example, if $s_{s\min} > s_g(t)$, then the source agent will not be able to achieve a speed slower or equal to the slower moving agent. As a result, the only option left for the faster moving agent is to speed up and overtake the slower moving agent.

Figure 7.17: Two-way pedestrian crossing.

7.4 More Complex Behaviours

In order to generate a realistic crowd, it is important that our agents are able to react differently in their own way when navigating through the environment. That is, real pedestrians should be able to choose how and when they want to avoid the collision. To this end, we will attempt to create heterogeneous groups of pedestrians by setting different speed constraints that represents different comfortable walking speeds for different agents. Note that there are sometimes more than one method of generating a behaviour using our micro-level navigation model through a combination of different constraints. We will describe how some of these behaviours can be generated using some of the constraints.

7.4.1 Two Groups

In this experiment, we have created five agents with different speed constraints, i.e., different minimum and maximum speeds. These agents also have the capability to generate collision avoidance constraints as in Section 4.1.2. They are divided into two groups represented by general direction that they are travelling. One of the groups is travelling from left to right and the

Figure 7.18: Three-way pedestrian crossing.

other group is travelling from right to left. This setup will ensure that each agent is required to avoid another agent at least once if not more than once to reach their destination. Figure 7.17 shows some key-frames capture of our experiment. Note that each agent is able to choose its own avoidance behaviour based on their comfortable walking speeds when it encounters a potential collision. For example, some agents are given a wide speed constraint ($0.5\text{ms}^{-1} \leq s_s(t) \leq 3\text{ms}^{-1}$) which reflects the fact that some pedestrians are willing to make short bursts of speeds to avoid a collision.

7.4.2 Three Groups

To make it even harder for the agents to avoid other agents and navigate to their destination, we increased the number of agents to seven and organised them into three groups. Figure 7.18 shows the screen captures from our experiment. Note that the agents are still able to navigate through the environment without colliding with other agents. Each agent is also able to adopt the a different collision avoidance strategy to avoid other agents based on their predetermined speed constraints representing different personalities. For example, an impatient agent can be represented with a high speed constraint and relative speed inflexibility ($1.5\text{ms}^{-1} \leq s_s(t) \leq 1.8\text{ms}^{-1}$).

Figure 7.19: Different social status.

7.4.3 Social Status

Social status represents the honor or prestige that is associated with one's standing in the society. Social status can be earned as a result of one's accomplishment or inherited due to one's birth rights. Due to the existence of social classes in most modern societies, different social behaviours may be appropriate for people with different social status [122]. According to sociologist Bourdieu, these behaviours may include their own social presentation and expectation of behaviours from other social groups [11]. Therefore, social status is an important consideration when modelling pedestrian flows in different societies. Specifically, pedestrians with higher social status may expect to have the *right-of-way* when moving through the environment, i.e., they will not expect to give way to groups with lower social status. Similarly, entourage of some important people such as city mayors, mafia bosses and celebrities are also expected to protect and follow the important person. These people will form groups around the important person and are not expected to give way to passing pedestrians.

The objective of this experiment is therefore to show that our agents are able to demonstrate navigation behaviours that reflect different social status.

Figure 7.19 shows the results of our experiment. The group of agents on the left is given specific instructions to ignore collision avoidance with agents not part of the group. In addition, an orientation constraint described in Section 4.1.4 is used to keep the group *around* the important person. On the other hand, the group of agents on the right are not given such instructions. Instead they have similar speed and collision avoidance constraints as the agents described in Section 7.3.1. We therefore see that the left group is able to navigate past the right group without being scattered while the right group is found to have scattered around the left group.

7.5 Dense Pedestrian Crowd Simulation

We have discussed about the simple pedestrian behaviours that our agents can demonstrate in Section 7.3, and demonstrated how different agent personalities can be demonstrated using our micro-level navigation model in Section 7.4. In this section, we will be using the extended micro-level navigation model that we have discussed in Section 6 in a dense pedestrian crowd so as to observe *emergent behaviours* such as *lane formation*. In particular, ad-hoc lanes (Section 7.5.1) has been observed to occur naturally without specifying additional constraints for the agents. Voluntary lanes, as we will describe in Section 7.5.2, can also be generated by adding orientation constraints to our agents. We will also discuss the role of *personal space* towards the generation of realistic pedestrian navigation through a dense crowd in Section 7.5.3.

Figure 7.20: Simulated Ad-hoc lanes.

7.5.1 Ad-Hoc Lane Formation

In a dense crowd, ad-hoc lane formation is characterized by several temporary lane-like agent movement through the environment. In our simulation, ad-hoc lanes can be formed using a combination of the *following* and *collision avoidance* behaviours as we have mentioned earlier. In particular, a lane is first formed when a leading agent decides use an *avoidance behaviour* to avoid the agent in front, hence *spearheading* a lane. Following agents then decide to reinforce the lane by following behind the lead agent rather than form a new lane themselves.

The purpose of our experiment is to test whether our agents are able to form ad-hoc lanes through *self-organisation* using our proposed micro-level navigation model. Speed constraints of the agent will determine whether the agent will follow a lane or create a new lane. Figure 7.20 shows the results of our experiments. Note that the agents were indeed able to *self-organise* into lanes without being explicitly programmed to do so. The lanes formed were also relatively short-lived as we have expected.

Figure 7.21: Voluntary lane formation at Batu Caves, Malaysia.

Figure 7.22: Simulated voluntary lane formation.

7.5.2 Voluntary Lane Formation

Unlike vehicular flows, pedestrian flows do not have *assigned* lanes to follow. At the same time, there are also no legal penalties for pedestrians who do not follow lanes. Yet these voluntary lanes are still being formed in places such as train stations (Figure 6.2) and crowded underground passes despite the lack of clearly marked lanes. Voluntary lanes are also frequently found on stairs (Figure 7.21) and escalators.

In this experiment, we attempt to recreate the voluntary lane formation behaviour using our micro-level navigation model. In particular, orientation

constraint coupled with priority ordering of the constraints, as we have discussed in Sections 4.1.4 and 4.1.6 respectively, is used to keep the agents in their own lane. Figure 7.22 shows the results of our experiment. Note that the lanes persisted throughout the simulation as though an invisible barrier had held them in place.

7.5.3 Effects of Personal Space

Figure 7.23 shows crowds of increasing density. Note that the space maintained by different pedestrians decreases as the density of the crowd increases. This suggests that pedestrians have a variable *personal space* that partially depends on the population density around the pedestrian. i.e., if there are more free space around the pedestrian, the pedestrians will tend to maintain a larger personal space. *Personal space* may also be affected by people's relationship, e.g., pedestrians tend to keep a smaller *personal space* with those that they know, such as friends and relatives. In this experiment, we will test these real-life observations with our agent's navigation model.

For the purpose of this experiment, we have created crowds of varying density and examined the results of changing the agent's personal space on the realism of the pedestrian simulation. We control the density of the agents in the area by increasing or reducing the rate at which new agents are introduced into the virtual environment. For example, we introduced one agent every ten frames to generate a low density pedestrian crowd. Similarly, we introduced one agent every five and one agent every frame to generate a medium and high density pedestrian crowd respectively.

From Figure 7.24, it can be noticed that the simulation generates more

Figure 7.23: Crowd with different density levels.

Figure 7.24: Effects of the agent's personal space.

realistic results for agents with larger *personal space* at lower density crowd (compare Figure 7.24(a) and 7.24d). On the other hand, agents with smaller *personal space* generates more realistic results for higher density crowds (compare Figure 7.24(c) and 7.24(f)). Therefore it will be appropriate to assume that real pedestrians do indeed change the radius of their personal space to suit the population density of the pedestrians around them, and it will be important to consider the effects of crowd density on the personal space of an agent.

7.6 Pedestrian Simulation in Complex Environments

Real pedestrians are not found only in narrow corridors and wide open areas, they are also found in many urban and indoor environments. However to simulate pedestrian flows in such areas, the macro-level navigation model will be required, i.e., the micro-level navigation model is by itself insufficient to guide an agent through a complex environment. Specifically, the macro-level navigation model's path-finding capability is required for the agent to find its way around a complex environment.

In this section, we will now demonstrate our the macro-level navigation model can be used in practice. We will show our agent's path-finding capability in Section 7.6.1 using an urban environment. Then in Section 7.6.2, we will demonstrate our agent's ability to avoid congestions using the heuristics discussed in Section 5.2.2. Finally in Section 7.6.3, we will describe a scenario where we have assigned different roles for different agents in the simulation to mimic a scene in a shopping mall.

7.6.1 Path-finding Capability

In this experiment, we have created an environment to represent a pedestrian zone in an urban area. The main objective of this experiment is to test the use of the macro-level navigation model for path-finding together with the use of the micro-level navigation model for steering. We will demonstrate two scenarios. In the first scenario, all agents are moving from left to right; and in the second scenario, agents are coming in from four different directions. The latter case is to increase the difficulty of navigating through the environment

Figure 7.25: One directional pedestrian flow.

Figure 7.26: Four directional pedestrian flow.

and understand its effects on our proposed macro-level navigation model.

Figure 7.25 shows the simulation results for the first scenario. Figure 7.26 shows the result for the second scenario. Note that the agents represented by the colored dots from this perspective are able to navigate around the static objects instead of passing right through them. This capability is important in pedestrian simulations with urban and indoor environments. Also note the relative chaos that has occurred in Figure 7.26 especially around the area where the agents from four different directions meet. In particular, the movement of the pedestrians are slowed in those areas, and hence reducing the efficiency of pedestrian movement. Similar observations have also been made by Teknomo who proposed the construction of simple pedestrian facilities, such as round-about, to help improve the efficiency of the pedestrian flows in such situations [113].

Figure 7.27: Use of detours to avoid congestions.

7.6.2 Congestion Avoidance

Congestion avoidance refers to a pedestrian's tendency to avoid crowded areas as they move around the environment. This means that if there is an easier route to the destination without going through the crowd, a pedestrian may choose that route even if the route distance may actually be longer. The choice to take detours depends may depend upon the pedestrian's personality and needs, and this choice can be reflected by the heuristics used in the path-planner. In particular, w_{pp} and w_{dp} in Equation 5.2, model a pedestrian's tendency to take detours to avoid congestions. For example, if an agent's w_{pp} is set to zero then the agent will never take detours to avoid a congestion.

Figure 7.27 shows a simple scenario where agents are able to distribute themselves among the two exits of the room to leave the room even when the right exit is further away. In order to achieve this effect, we randomly assign the values of w_{pp} and w_{dp} such that $w_{pp} + w_{dp} = 1$ for each agent. As the congestion on the left (nearer) exit starts to clear off, the remaining pedestrians chose the left exit to leave the room as it results in a shorter path. Figure 7.28 shows how some of the pedestrians in a shopping malls are able to take small detours to prevent the need of navigating through a dense

Figure 7.28: Congestion avoidance in a shopping mall.

crowd.

7.6.3 Shopping Mall

In this experiment, we have assigned different roles to different agents in the simulation. In particular, the role that an agent can take is either a store assistant, a shoppers or a pedestrian. The objective of this experiment is an attempt to model a real-life scenario using our proposed pedestrian navigation model and observe the pedestrian behaviours that are generated. Store assistants are only allowed to move within their own store and attend to shoppers that have visited their store. Shoppers will stay within the shopping mall and visit the different stores. Pedestrians are just passing through the area, entering from one entrance of the mall and leaving from the other entrance. Figure 7.29 shows a screen capture of the experiment.

The store attendant role can be realised by giving those agents a partial set of the abstract map that describes the environment for path finding (Section 5.2.1). This means that the agents will only be able to move around

Figure 7.29: Agents with different roles in a shopping mall.

that subset of the environment representing the stores hence behaving like store attendants. The shopper role is realised by giving those agents a list of waypoints representing the locations of different stores. When finding a new destination, the agent will choose from a list of shops known to the agent hence achieving a shopper's behaviour. Similarly, pedestrians are given destination lists containing only the entrance and exits of the area. This means that the pedestrians will only move between the two exits of the area and hence generating a *passing-through* behaviour. This experiment therefore demonstrates the versatility of our pedestrian navigation model and the importance of personal perception about the environment in the generation of

realistic pedestrian behaviours.

7.7 Real-Time Performance Analysis

The mostly commonly used performance indicator for a real-time simulation is the render period of each simulation frame. In particular, the render period is compared with the number of agents in the environment to determine the efficiency of the navigation model. However, this indicator is not enough in our case to truly understand the performance of our simulation. This is because many factors, both internal and external may affect our simulation's performance. These factors may include the complexity of the environment, the density of the crowd and the number of memory operations required for dynamic memory allocations. Therefore it will be more accurate to observe the changes in frame rate over simulation time than over the number of agents. In addition, the variations in frame rates over simulation time is another important performance indicator in our case. In particular, large variations in render period will result in noticeable lags in the simulation, hence reducing the smoothness of the agent's animation.

In this section, we will analyse the performance of our simulation using two scenarios. They are the one-directional and four-directional pedestrian flows in a built up area as described in Section 7.6.1. These scenarios are used to benchmark our simulation because they make use of both the micro-level and macro-level navigation model and operates in a complex environment. In order to better measure the performance of our simulation, we also need to understand the key computation costs of the simulation.

First of all, each agent needs to perform a scan of the environment during each frame. Therefore the agent count will be a sufficient indicator for the computational costs for vision sampling because the same action is performed by every single agent. Secondly, each agent need to perform collision avoidance between itself and other agents in the environment. The total number of collision avoidance operations performed in a single frame will change depending on the density of pedestrians in the environment, i.e., not all agents need to perform collision avoidance at every frame. Specifically, if there are no predicted collisions and the agent is already on course to its destination, then there is no need to compute new steering parameters for the agent. Therefore, sparse crowds will require less collision avoidance while denser crowds will require more. As a result, sparser crowds will result in a lower render period than denser crowds.

Finally, each agent needs to plan its path using the macro-level navigation model. Each agent in the environment needs perform this action at least once during its lifetime. Hence we are primarily concerned with path replans as a result of collision avoidance with another agent and its effect on the render period of the simulation. Usually the micro-level navigation model is capable of bringing an agent back on course, however there are situations where the collision avoidance behaviour brings the agent so far away from its course that a path replan is necessary. This is more likely to occur in a dense crowd than in a sparse crowd.

Therefore we have chosen three parameters, the total number of collision avoidance performed by all agents in the current frame, the total number of path replans performed by all agents in the current frame, and total number

of agents in the current frame. We will describe the effects of each of these indicators on the overall performance of the simulation in this section.

7.7.1 One-Directional Pedestrian Flow

In this experiment, we have created a one-directional pedestrian flow in our build up area described in Section 7.2.4. The objective is to analyse the changes in render period at different points of the simulation. We also want to estimate the number of agents that can be supported for the simulation to run at a reasonable frame rate. We will control the density of the crowd by controlling the rate at which new agents are created. For this experiment, one new agent is created every twenty frames.

Figure 7.30 shows the variations in number of collision avoidance and path-replans performed by all agents in a single frame, over time for a one directional pedestrian flow. Note that both numbers are relatively low for a one directional pedestrian flow. The number of agents in the environment is also plotted on the secondary axis. Note also that the path replans are relatively infrequent as compared to the number of avoidance behaviours executed. This means that the most of the computations are performed by micro-level navigation model. Hence the performance of the simulation will depend significantly upon the computational efficiency of the micro-level navigation model. This trend appears to be the same for all our pedestrian simulations.

Figure 7.31 shows the variation in render period with the changes in the number of agents over time. The increase is almost linear with the increase in number of agents. For the simulation to run at around 20 frames per second

Figure 7.30: Agent behaviour in one directional pedestrian flow.

(fps), the render period has to be between 40-60 ms ($1000/\text{fps}$). This means that the simulation is able to handle around 80-100 agents at this frame rate.

7.7.2 Four-Directional Pedestrian Flow

In this experiment, we created four directions of pedestrian flows and analysed the effect on the simulation's render period. Pedestrians are created at the rate of one pedestrian every twenty frames.

Figure 7.32 shows a period of high collision avoidance activity just before the number of agents reaches its peak. This means that agents from all the four flows have met, resulting many collision avoidance behaviours to be performed. Eventually, the situation stabilises as new lanes are formed and less collision avoidance between agents are needed. Note that the number of

Figure 7.31: Render period for one directional pedestrian flow.

collision avoidance and path replans is generally higher than a single directional pedestrian flow. This justifies why real pedestrians need to form lanes to reach their destination faster.

Figure 7.33 shows a similar increase in render period just before the number of agents is peaked. The reason as we have explain is due to the increased collision avoidance behaviour among the pedestrians. Note that the variations in render period is relatively large. Our simulation is however able to dampen the effect of these variations by adopting a frame-rate depended update cycle, i.e., agents are moved forward according to how much time has elapsed since the last update cycle. Nevertheless, the simulation was not as smooth as it could be and we found that most of these operations were actually due to memory operations. In particular, memory allocations

Figure 7.32: Agent behaviour in four directional pedestrian flows.

and deallocation operations are one of the key reasons for the frame rate variations.

7.8 Performance Optimisation Using Free-lists

Free-list is a method of memory management that speeds up the memory allocation and deallocation process using prior knowledge about the size of the memory requests. As a result, free-lists are very useful in optimizing the performance of a crowd simulation due the presence of many same size data structures used to represent the crowd in the simulation.

We have implemented free-lists for our crowd simulation by overwriting the default **new** and **delete** operations for memory allocation and deallocation in C++. Table 7.1 shows the average render period for each agent in the

Figure 7.33: Render period for four directional pedestrian flows.

one and four directional pedestrian flows as described in Section 7.6.1. We have also compared the render period between the different scenarios using free-list assisted memory management. Based on the average render periods from the different scenarios, we can conclude that the used of free-list assisted memory management has resulted in a performance improvement of more than 15%.

We will now give the detailed results of our experiments. In Section 7.8.1, we have performed a simple case study to illustrate the effectiveness of using free-lists for path-planning. Then we will present and compare the performance difference of our simulation with and without the use of free-lists in Sections 7.8.2 and 7.8.3

Scenario	Render Period (ms)	Improvement (%)
One Directional Flow w/o Freelists	0.500024069	
One Directional Flow w Freelists	0.415485952	17.00%
Four-Directional Flows w/o Freelists	0.533407158	
Four-Directional Flows w Freelists	0.434616326	18.45%

Table 7.1: Performance improvement using free-lists.

7.8.1 Free-Lists for Path-Planning

In this experiment, we will illustrate the performance improvement of using free-lists in path-planning. Path-planning is used as a case study because there is significant number of dynamic memory allocation and deallocation required to complete the path-planning algorithm. In particular, dynamic memory allocation is needed to expand the open and closed list of the A* algorithm to mark the candidate graph nodes for visitation.

We have used the urban built up environment as the benchmark for this experiment. The environment is prepared for path-finding using different number of waypoints. In particular, the environment will have four different waypoints graphs containing different number of waypoints. The time required to find a route from one end of the environment is then taken and repeated for three times. The average of the three samples then represents the average time for an agent to plan a path between the two locations. The experiment is first performed using only the operating system's memory manager and repeated using free-lists.

Table 7.2 summarizes the results from multiple runs of the experiment.

No. Waypoints	64	240	420	1040
Operating System (ms)	0.811	1.324	3.158	11.642
Freelists (ms)	0.375	0.552	1.221	2.878
Improvement	1.16	1.4	1.59	3.05

Table 7.2: Performance improvements for path-finding.

Note that when the environment is complex with up to 1000 waypoints, it can take around 10 ms to compute a path. This means the performance of the simulation will be significantly affected for each path replan that an agent makes in such an environment. The use of free-lists helps to reduce the computational cost for path-planning to 3 ms.

7.8.2 One-Directional Pedestrian Flow

In this experiment, we created a one-directional pedestrian flow using the same setup as Section 7.7.1. The only difference is the use of free-list to improve the performance of the simulation. Specifically, free-lists are used to manage the memory for creating agent constraints, creating agent 3D models, creating agent visibility list of objects and for path-finding. We observed that there is a significant improvement in the performance of the simulation as a result of using free-lists. For example, to maintain the render period between 40-60 ms, we are able to control and render between 110 and 130 agents using free-lists as compared to between 80 and 100 agents without the use of free-lists.

Figure 7.34 shows the number of actions taken by each agent throughout the simulation. Note that the number of collision avoidance and path replans are not affected by the implementation of a free-list. Instead, they

Figure 7.34: Agent behaviour using freelists.

are presented here to better explain the changes in the render period of the simulation over time. Comparing Figure 7.35 and 7.31, we can observe not only the simulation runs faster but there is also significantly less variation in the frame rate. This will allow us to create a smoother running simulation as well as having more agents in the simulation concurrently.

7.8.3 Four-Directional Pedestrian Flow

Using a free-list aided memory management, we have created a four pedestrian flow scenario using the same setup described in Section 7.7.2. Figure 7.36 shows the number of actions taken by the pedestrians in a four-directional pedestrian flow to avoid both structural obstacles and other agents. Again, agent behaviour is not affected by the use of free-lists but are pre-

Figure 7.35: Render period using freelists.

sented to explain the changes in frame rate with the number of agents.

Figure 7.37 shows the change in render period with the change in number of agents in the environment. When compared with Figure 7.33, we can also observe an increase performance of the simulation and a much more stable frame rate throughout the simulation. For example at maximum load, the simulation is able to handle around 275 agents with a render period of 200ms using free-list. This is considered an improvement as compared to 250 agents with a render period of 250ms without the use of free-lists. This is a desirable trait because it means that the simulation is capable of supporting more agents during run-time and also achieve a smoother animation.

Figure 7.36: Agent behaviour using free-lists.

Figure 7.37: Render period using free-lists.

7.9 Performance Optimisation Using Time Slicing

In Section 7.8, we have discussed a method of optimising the performance of the simulation by increasing the efficiency of the memory allocation process. In this section, we will examine the effects of another performance optimisation strategy. We propose to reduce the total number of actions that are taken by all agents in a single frame to improve the performance of the simulation. This can be accomplished by separating the agents into *buckets*.

These buckets are each processed in sequence and processing is limited to one bucket per frame. If the agent belongs to the same bucket being processed, then the agent is allowed to perform a scan of the environment, formulate collision avoidance constraints, perform path-replans and compute new steering parameters. Agents that do not belong to the bucket being processed are not allowed to make changes to their speed and orientation, instead they will just update their position using the same speed and orientation from their previous update cycle. This divide and conquer approach has gained us a reduction in the simulation's render period with some trade-off.

In Section 7.9.1, we will present our experiment data as a result of using different number of buckets on a four-directional pedestrian flow using a similar setup. Then, in Section 7.9.2, we will discuss about the trade-off associated with the use of buckets. Finally, we will discuss the potential use of buckets at lower levels of detail by comparing the images captured from the simulation in Section 7.9.3.

Figure 7.38: Agent behaviour using two buckets.

7.9.1 Using Buckets to Improve Performance

In this experiment, we will progressively increase the number of buckets to examine its effect on the number of actions performed by agents during each frame and its effect on the render period of the simulation. We have used the four-way pedestrian flow scenario described in Section 7.8.3 for this experiment. Free-list aided memory management is also used in all the experiments.

Comparing Figure 7.38 and Figure 7.36, we can observe a reduction in agent activity level. For example, there were 75 collision avoidance actions taken for 275 agents as compared to 65 collision avoidance actions taken for 350 agents. We will provide a more detailed comparison between about the reduction in agent activity and what it means to the simulation in Section

Figure 7.39: Render period using two buckets.

7.9.2. Similarly, we can observe that the simulation runs faster by comparing Figure 7.39 and 7.37. For example, at peak load, the simulation is capable of controlling and rendering 350 agents with a render period of around 150ms when using two buckets. This is as compared to 275 agents with estimated render period of 200ms when no buckets are used.

We further increased the number of buckets to four and as shown in Figure 7.40 further reduces agent activity level when compared to Figure 7.38. For example, there is now only 50 collision avoidances for 350 agents as compared to 65 collision avoidances for 350 agents. At the same time, the reduction in activity allows more agents to be controlled during run time by the simulation. For example, the simulation can handle 450 agents with a render period of approximately 125ms at maximum capacity.

Figure 7.40: Agent behaviour using two four buckets.

Figure 7.41: Render period using four buckets.

Figure 7.42: Agent behaviour using eight buckets.

Figure 7.43: Render period using eight buckets.

Finally, when we increased the number of buckets to eight, we again found a performance boost to the simulation. Figure 7.42 shows our agent's activity level when eight buckets were used. As with previous test cases, it shows a reduction in agent activity level. Furthermore, Figure 7.43 shows that the simulation is able control 500 agents using an average render period of 100 ms. As compared to Figure 7.37, the simulation needs more than 100 ms to render 150 agents using only free-lists and no time slicing mechanisms.

7.9.2 Trade-Offs For Using Time-slicing

From our experiments in Section 7.9.1, we can expect that agent activity level will continue to reduce if we increase the number of buckets further. The result will therefore be a continuous improvement in the performance of the simulation. In this section, we will examine more closely what this reduction in agent activity level means and how it will actually affect the output of the simulation.

Figure 7.44 shows a degree six polynomial trend line that can be used to compare the render period for the simulation using different number of buckets. The performance indicator for one-bucket is obtained from the results of the four-directional pedestrian flow simulation using free-lists (Section 7.8.3). As seen from Figure 7.44, a direct comparison between the different schemes is difficult because being a simulation, the distribution of agents in the environment cannot be compared across different runs. However it can be seen that the simulation generally has a lower render period when eight buckets are used.

Table 7.3 shows the tabulated average of the agent's activity level as well

Figure 7.44: Render period using different number of buckets.

as render period. These figures are computed based on the average activity of all agents in the environment for the entire duration of the simulation. For example, when the number of buckets is equal to four, each agent has a 0.58% chance of replanning a path at a frame and a 6.83% chance of execution a collision avoidance behaviour with another agent at the same frame. As compared to a 1.02% chance of path replanning and a 14% chance executing

Buckets (count)	Average Render Period (ns/agent-frame)	Path Replans (per agent-frame)	Collision Avoidance (per agent-frame)
1	434.6	0.0102	0.1400
2	292.2	0.0084	0.1013
4	190.3	0.0058	0.0683
8	108.9	0.0027	0.0346

Table 7.3: Trade-offs of Time-slicing.

Buckets (count)	Renderable Agents (N times)	Wall Collision (N times)	Agent Collision (N times)
2	1.48	1.21	1.38
4	2.28	1.76	2.04
8	3.99	3.78	4.04

Table 7.4: Agent behaviour and Time-slicing.

a collision avoidance behaviour when using one bucket (or no time-slicing), the differences represents *missed chances* of avoiding a structural obstacle or other agents, i.e., collisions that occurred as a result of agents being unable to update their steering parameters every frame. These results can be better explained using Table 7.4.

Table 7.4 uses the four-directional pedestrian flow with free-list scenario as described in Section 7.8.3 as the benchmark to represent that no collision avoidance will be missed by an agent. Then the results can be interpreted that the simulation can handle 1.48 times more agents when using two buckets than when one bucket is used. At the same time, the agent is 1.21 times more likely to collide with a wall and 1.38 times more likely to collide with another agent due to the use of time-slicing. Similarly, when eight buckets are used, the simulation can handle 3.99 times more agents than it will otherwise be possible but agents are 3.78 times more likely to collide with a wall and 4.04 times more likely to collide with another agent. Figure 7.45 will give a better illustration of what these numbers mean to the output of the simulation.

Figure 7.45 shows a representative capture of the simulation where different number of buckets are used. Figure 7.45(a) shows a capture of the

Figure 7.45: Improving performance using Time-slicing.

simulation when time-slicing is not used. Figure 7.45(b) uses two buckets and some agents begin to show *clustering* as the agents starts missing collision avoidance with another agents. Some wall collision and more agent clustering can also be seen in Figure 7.45(c) with four buckets and 7.45(d) with eight buckets.

7.9.3 Time-Slicing for Different Levels of Detail

Although there are problems with the use of time slicing such as missed collisions with other agents, the performance improvement gained from time slicing cannot be ignored. In particular, time slicing can be used when the camera moves between different zoom levels. Referring to Figure 7.46, we can

Figure 7.46: Time-slicing and Level of Detail.

observe that the actual collision between the agents and the wall is no longer obvious. This suggests that a dynamic time-slicing approach dependant on the position of the camera can be used to improve the performance of the simulation. Secondly, wall collision due time-slicing can be reduced by creating a larger bounding volume around the structural obstacles. The bounding volume will then be used during the construction of the waypoints for the environment instead of the actual geometry of the obstacle. As a result, this may prevent the agents from navigating too close to the wall and provide sufficient time for the agent to avoid collision with the wall even when using time-slicing.

7.10 Summary

We have discussed our implementation of various support modules for the pedestrian simulation. In particular we have used the A7 Game Studio to aid the development of our pedestrian simulation. We have implemented four different environments for the simulation, each representing a type of real-life environment. Each of these environments is suitable for visualizing a different type of pedestrian behaviour.

We have demonstrated various basic pedestrian behaviours such as *following* and *overtaking*. We have also explained how these behaviours can be used as basic building blocks for more complex behaviours like *lane formation*. All the navigation behaviours demonstrated by our agents are *emergent* due to the fact that no explicit instructions were give to the agent to for example form lanes. Instead the agent's navigation behaviours were merely constrained to avoid collision.

We have shown how various real-life pedestrian behaviours like entourage movements and voluntary lane formation can be generated by our micro-level navigation model. In particular, different constraints can be added to the agent to represent the agent's personality and habit. We have also tested our agent's path-finding capability and the agent's ability to use both the macro-level and micro-level navigation model to find its way around complex environments. In the shopping mall scenario, we have also demonstrated that our pedestrian model can be easily modified to give our agents different roles.

We have analysed the real-time performance of our pedestrian simula-

tion and explored two strategies for optimising the simulation's performance. Free-list has been implemented to optimise the memory allocation and deallocation process of our pedestrian simulation. We found that free-list is able to improve the performance of the simulation by more than 15% during the simulation of four-directional pedestrian flows in a complex urban environment. We have also explored the use of time-slicing to reduce the agent's activity level in a single frame. Time slicing can especially useful in speed up the performance of the simulation with different level of details.

8 Conclusion

The modelling and simulation of realistic pedestrian behaviour faces many challenges ranging from behavioural modelling to crowd simulation. Realistic behaviours not only make the simulation more believable but will also make the results and predictions from the simulation more accurate. In particular, a microscopic pedestrian simulation can be used for predicting the evacuation time for egress simulations, studying the adequacy of pedestrian facilities under different situations and automatic generation of pedestrian crowd of believable animation.

In this thesis, the problem is approached by first observing the navigation behaviours of real pedestrians and the understanding the various factors that affect the navigation decisions of a pedestrian. Based on these observations, a realistic representation of an agent for a pedestrian and a two level navigation model has been proposed. In particular, the agent's navigation model consists of a macro-level navigation model and a micro-level navigation model. A two level navigation model is used because such an approach is able to mimic the human-decision making process for moving through a highly dynamic environment.

In the macro-level navigation model, we have proposed the use of individual perception about the environment for path-finding and demonstrated how this can lead to more realistic path-planning behaviours. Based on this individual perception, an abstract path is then generated for the agent to follow. The generated path can reflect different pedestrian habits and individual ignorance about the environment. The proposed micro-level naviga-

tion model considers physical and psychological factors affecting pedestrian movement as constraints on an agent's steering parameters. Such an approach allows considerations from unrelated domains such as patience and age to be effectively combined together for the generation of the agent's movement.

However, increasing the realism of agent's behaviour usually means an increase in computational complexity. In order to influence the composition of a the pedestrian crowd, and to receive fast feedback on how a factor may affect the movement of agents in the environment, it is necessary to seek more efficient methods of computing the agent's navigation behaviours. For that matter, it may sometimes be necessary to forgo a more accurate representation of an agent with a more computationally efficient representation. Bearing that in mind, two optimisation techniques have also been proposed to improve the *overall* performance of the simulation. They are memory management and time-slicing which we have discussed in Sections 7.1 and 7.9 respectively.

For the remainder of this chapter, we will summarise the contributions of this thesis in Section 8.1 and discuss the possible future extensions to our work in Section 8.2.

8.1 Contributions

The main contributions of this thesis are as follows:

1. We have proposed a realistic model to represent a real pedestrian with human-like sensory inputs, memory and attention systems;

2. We have proposed a two-tier navigation model for our agents to navigate in different types of environment ranging from wide open spaces to urban build up areas;
3. We have proposed a novel micro-level navigation model for our agents that is capable of demonstrating realistic individual navigation behaviours like collision avoidance and overtaking;
4. We have demonstrated how lanes can be formed in a dense pedestrian crowd using our micro-level navigation model. In particular, the differences between ad-hoc and voluntary lane formation has been explored. The lane-formation behaviour of our agents has also proven to be emergent behaviours rather than as a result of an explicit set of instructions.
5. We have proposed a macro-level navigation model that is able to compute a rough path for the agents to follow. Much like a real-pedestrian, navigation details like collision avoidance are only handled by the micro-level navigation model. By introducing personal habits and experiences into the path-finding process, we are able to generate personalised paths for the agents. In particular, we are able to demonstrate congestion avoidance behaviours using the macro-level navigation model.
6. We have proposed a memory management mechanism targeted at reducing the computational overhead for dynamic memory requests to improve the performance of our navigation model in a crowd scenario. We have proposed the use of time-slicing to further improve the performance of our pedestrian simulation. To this end, we have presented

the trade off for using time-slicing on the realism of the pedestrian simulation. We also reasoned that time-slicing may be a good strategy for improving the performance of the simulation depending on the level of detail required from the simulation.

7. To evaluate the effectiveness of using free-lists for memory management, we have designed experiments to test the render periods for the agents using free-lists for memory management. We found that the overall performance of the simulation can be increased by up to 18% if free-lists are used. To understand the challenges of simulating realistic pedestrian behaviours, we have implemented different virtual environments and tested our agent's ability to move realistically in these environments. In particular, these environments were chosen to represent different real-time scenarios;

In short, we have demonstrated how realistic navigation behaviours for microscopic pedestrian simulation can be achieved within the constraints of real-time simulation.

8.2 Future Challenges

The computational powers of our computers will increase over time and this allows for more and more complex behaviours that can be generated by autonomous agents. Therefore, future challenges for realistic pedestrian simulation will be centered on developing even more realistic agents that mimic the behaviours of a real human. We therefore raise the following challenges which we believe are interesting future works;

1. Additional constraints can be derived to provide the agent with a greater vocabulary of navigation behaviours. In particular, the handling of pedestrian behaviours in a dense crowd simulation such as queuing and pushing.
2. Our agents only have a single channel of animation, the entire body of the agent itself. Real pedestrians however are able to turn their heads and bodies when navigating through the environment. We believe this can be achieved by finding control parameters that will enable the control of individual components of the agent's 3D model, e.g. separate head and body coordination.
3. We have listed several interesting pedestrian behaviours in a dense pedestrian crowd. Behaviours such as queuing and pushing are important behaviours necessary to increase the realism of a dense pedestrian crowd simulation.
4. Crowd scenes are increasingly common in computer generate imagery (CGI) for movies and animation. We believe that an automatic generation of a realistic pedestrian crowd will have many applications in that area.
5. Finally, pedestrian navigation behaviours constitute the most basic functionality of autonomous agents. However, autonomous agents who are merely able to move in a realistic manner will not have as many applications as autonomous agents who are able to perform other even more interesting behaviours such as tactical movements. We believe

that such behaviours are possible by further extending our proposed navigation module.

References

- [1] *A6 Game Engine*. <http://www.3dgamestudio.com/>.
- [2] S. Al-nasur and P. Kachroo. A microscopic-to-macroscopic crowd dynamic model. In *IEEE Conference on Intelligent Transport Systems 2006, ITSC '06*, pages 606–611, Toronto, Canada, 2006.
- [3] I. Altman. *The Environment and Social Behavior: Privacy, Personal Space, Territory, and Crowding*. Brooks/Cole Publishing Company, 1975.
- [4] M. Asada. Map building for a mobile robot from sensory data. *IEEE Transactions on System, Man and Cybernetics*, 20:1326–1336, 1990.
- [5] A. Baddeley. *Human Memory: Theory and Practice*. Psychology Press, 1997.
- [6] J. Bailenson, J. Balscovich, A. Beall, and J. Loomis. Equilibrium theory revisited: Mutual gaze and personal space in virtual environments. *Presence: Teleoperators and Virtual Environments*, 10:583–598, 2001.
- [7] J. Balasuriya, C. Marasinghe, and W. Keigo. Anfis for adaptive personal space determination for ubiquitous robots. In *Proceedings of 7th IEEE International Conference on Computer and Information Technology*, pages 973–978, Aizu-Wakamatsu, Japan, 2007.
- [8] D. Bauer, D. Seer, and N. Brndle. Macroscopic pedestrian flow simulation for designing crowd control measures in public transport after special events. pages 1035–1042, San Diego, USA, 2007.
- [9] C. Beaulieu. Intercultural study of personal space: A case study. *Journal of Applied Social Psychology*, 34:794–805, 2004.
- [10] V. J. Blue and J. L. Adler. Cellular automata microsimulation for modelling bi-directional pedestrian walkways. *Transportation Research Part B*, 35:293–312, 2001.

- [11] P. Bourdieu. *Distinction: A Social Critique of the Judgement of Taste*. Harvard University Press, 1984.
- [12] D. M. Bourg and G. Seeman. *AI for Game Developers*. O'Reilly, 2004.
- [13] C. M. Breder. Studies on social groupings in fishes. *Bulletin of the American Museum of Natural History*, 117:397–481, 1959.
- [14] D. Brogan and N. Johnson. Realistic human walking paths. In *Proceedings of 16th International Conference on Computer Animation and Social Agents 2003, CASA '03*, pages 94–101, 2003.
- [15] R. Brooks. Coherent behaviour from many adaptive processes. In *Proceedings of From Animals To Animats 3: SAB'94*, pages 22–30, Brighton, U.K., 1994.
- [16] C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295:507–525, 2001.
- [17] L. Celine, D. Marchal, and A. Meyer. Intuitive crowd behaviour in dense urban environment using local laws. *Theory and Practice of Computer Graphics '03*, pages 122–129, 2003.
- [18] S. Chopra-Khullar and H. Badler. Where to look ? automating attention behaviours for visual human characters. *Autonomous Agents and Multi-Agent Systems*, 4:9–23, 2001.
- [19] T. Conde and D. Thalmann. An artificial life environment for autonomous virtual agents with multi-sensorial and multi-perceptive features. *Computer Animation and Virtual Worlds*, 15:311–318, 2004.
- [20] F. Dapper, E. Prestes, M. Idiart, and N. L.P. Simulating pedestrian behaviour with potential fields. *Advances in Computer Graphics*, 4035:324–335, 2006.

- [21] M. Dickheiser. *AI Game Programming Wisdom 2*, chapter Inexpensive Precomputed Pathfinding Using a Navigation Set Hierachy, pages 103–114. Charles River Media, 2004.
- [22] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
- [23] C. Dyer, A. Rosenfeld, and H. Samet. Region representation: Boundary codes from quadtrees. *Communications of the ACM*, 23:171–179, 1980.
- [24] D. Eberly. *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics, 2nd Edition*. Morgan Kaufmann Publishers, 2007.
- [25] C. Ericson. *Real Time Collision Detection*. Morgan Kaufmann Publishers, 2006.
- [26] W. Fang, L. Yang, and W. Fan. Simulation of bi-direction pedestrian movement using a cellular automata model. *Physica A: Statistical Mechanics and its Application*, 321:633–640, 2003.
- [27] F. Feurtey. Simulating the collision avoidance behaviour of pedestrains. Master’s thesis, School of Engineering, University of Tokyo, 2000.
- [28] R. Floyd. Algorithm 97: Shortest path. *Communications of the ACM* 5, 6:345, 1962.
- [29] M. Gardner. Mathematical games: The fantastic combinations of john conway’s new solitaire game ‘life’. *Scientific American*, 223:120123, 1970.
- [30] C. Gloor, D. Cavens, E. Lange, K. Nagel, and W. Schmid. A pedestrian simulation for very large scale applications. In *A. Koch und P. Mandl (Hrsg.): Multi-Agenten-Systeme in der Geographie 2003*, 2003.
- [31] A. Godara, S. Lassarre, and A. Banos. Simulating pedestrian-vehicle interaction in an urban network using cellular automata and multi-agent models. *Traffic and Granula Flow ’05*, pages 411–418, 2005.

- [32] S. Goldstein et al. Scalable nonlinear dynamical systems for agent steering and crowd simulation. *Computer and Graphics*, 25:983–998, 2001.
- [33] R. Golledge. *Wayfinding Behavior: Cognitive Mapping and other Spatial Processes*. JHU Press, 1999.
- [34] M. Grimani. *Wall Building for RTS Games*, pages 425–437. Charles River Media, 2004.
- [35] J. Gutmann, M. Kukuchi, and M. Fukita. A floor and obstacle height map for 3d navigation of a humanoid robot. In *Proceedings of the IEEE International Conference on Robotics and Automation 2005, ICRA '05*, pages 1066–1071, Barcelona, Spain, 2005.
- [36] S. Gwynne et al. *Escape as a Social Response*. Society of Fire Protection Engineers, Boston, USA, 1999.
- [37] M. Habib and H. Asama. Efficient method to generate collision free paths for an autonomousmobile robot based on new free space structuring approach. In *In proceedings of International Workshop on Intelligent Robots and Systems '91*, volume 2, pages 563–567, Osaka, Japan, 1991.
- [38] M. Haciomeroglu, R. Laycock, and A. Day. Distributing pedestrians in a virtual environment. In *Proceedings International Conference on Cyberworlds 2007, CW '07*, pages 152–159, Hannover, Germany, 2007.
- [39] E. Hall. *The Hidden Dimension*. Anchor, 1990.
- [40] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics SSC4*, 2:100107, 1968.
- [41] D. Helbing. Collective phenomena and states in traffic and self-driven many particles systems. *Computation Material Science*, 30:180–187, 2004.
- [42] D. Helbing et al. *Simulation of Pedestrian Crowds In Normal and Evacuation Situations*. Springer, 2002.

- [43] D. Helbing et al. Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. *Transportation Science*, 39:1–24, 2005.
- [44] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical Review*, 51:4282–4286, 1995.
- [45] P. Herrero et al. Modelling the sensory abilities of intelligent virtual agents. *Autonomous Agents and Multi-Agent Systems*, 11(3):361–385, 2005.
- [46] R. Hill. Perceptual attention in virtual humans: Towards realistic and believable gaze behaviours. In *Proceedings of the AAAI Fall Symposium on Simulating Human Agents*, pages 46–52, Menlo Park, USA, 2000.
- [47] S. P. Hoogendoorn and P. H. L. Bovy. Gas-kinetic modeling and simulation of pedestrian flows. *Journal of the Transportation Research Board*, 1710:28–36, 2000.
- [48] S. P. Hoogendoorn and P. H. L. Bovy. Generic gas-kinetic traffic systems modeling with applications to vehicular traffic flow. *Transportation Research Part B*, 35:317–336, 2001.
- [49] S. P. Hoogendoorn and P. H. L. Bovy. Pedestrian route-choice and activity scheduling theory and models. *Transportation Research Part B*, 38:169–190, 2004.
- [50] W. H. Huang et al. Visual navigation and obstacle avoidance using a steering potential function. *Robotics and Autonomous Systems*, 54(4):288–299, 2006.
- [51] R. L. Hughes. A continuum theory for the flow of pedestrians. *Transport Research Part B*, 36:507–535, 2002.
- [52] M. Isobe, D. Helbing, and T. Nagatani. Many-particle simulation of the evacuation process from a room without visibility. *Physical Review Part E*, 69, 2004.
- [53] M. Johnstone and P. Wilson. The memory fragmentation problem: Solved ? *ACM SIGPLAN Notice*, 34:26–36, 1999.

- [54] U. Kelin, T. Schulze, and S. Straburger. Traffic simulation based on high level architecture. In *Proceedings of the 30th Conference on Winter Simulation*, pages 1095–1104, Washington, D.C., USA, 1998.
- [55] T. Kent. chapter Multi-tiered AI Layers and Terrain Analysis for RTS Games, pages 447–456. 2004.
- [56] Y. Kim, M. van Velsen, and R. Hill. Modelling dynamic perceptual attention in complex virtual environments. *Intelligent Virtual Agents*, 3661:266–277, 2005.
- [57] A. Kirchner. Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. *Physica A: Statistical Mechanics and its Applications*, 312:260–276, 2002.
- [58] A. Kirchner, K. Nishinari, and A. Schadschneider. Friction effects and clogging in a cellular automata model for pedestrian dynamics. *Physical Review E*, 67:1–10, 2003.
- [59] K. Knowlton. A fast storage allocator. *Communications of the ACM*, 8:623–625, 1965.
- [60] D. Knuth. *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. 1973.
- [61] S. Koenig and M. Likhachev. D* lite. In *Proceedings of the 18th National Conference on Artificial Intelligence '02, AAAI '02*, pages 476–483, Edmon-ton, U.S.A, 2002.
- [62] S. Koenig, C. Tovey, and Y. Smirnov. Performance bounds for planning in unknown terrain. *Artificial Intelligence*, 147:253–279, 2003.
- [63] W. L. Koh and S. P. Zhou. An extensible collision avoidance model for realistic self-driven autonomous agents. In *Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Appli-cations*, pages 7–14, Crete, Greece, 2007.

- [64] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation 1999*, volume 2, pages 1398–1404, Scaramento, USA, 1991.
- [65] R. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.
- [66] R. Korf. Real-time heuristic search. *Artificial Intelligence*, 42:189–211, 1990.
- [67] T. Lakoba, D. Kaup, and N. Kinlestein. Modifications of the helbing-molnfarkas vicsek social force model for pedestrian evolution. *Simulation*, 81:339–352, 2005.
- [68] F. Lamarche and S. Donikian. Crowd of virtual humans: A new approach for real-time navigation in complex and structured environments. *Computer Graphics Forum*, 23(3):509–518, 2004.
- [69] D. Le, B. R., and D. Thalmann. Integrating age attributes to virtual human locomotion. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Science*, XXXIV(5), 2003.
- [70] M. Lees, B. Logan, and G. Theodoropoulos. Agents, games and hla. *Simulation Modelling Practice and Theory*, 14:752–767, 2006.
- [71] L. Lidn. *Game Programming Gems 2*, chapter Strategic and Tactical Reasoning with Waypoints. Charles River Media, 2002.
- [72] B. Logan and G. Theodoropoulos. The distributed simulation of multi-agent systems. In *Proceedings of IEEE Journal, Special Issue on Agent-Oriented Software Approaches in Distributed Modelling and Simulation.*, volume 89, pages 174–186, UK, 2001.
- [73] M. Lozano, P. Morillo, et al. A distributed framework for scalable large-scale crowd simulation. *Virtual Reality*, 4563:111–121, 2007.

- [74] J. Manzo. Social control and the management of 'personal' space in shopping malls. *Space and Culture*, 8:83–97, 2005.
- [75] A. D. May. *Traffic Flow Fundamental*. Prentice Hall, 1990.
- [76] C. Miles and S. J. Louis. Towards the co-evolution of influence map tree based strategy game players. In *Proceedings of IEEE Symposium on Computational Intelligence and Games 2006*, pages 75–82, Reno, NV, U.S.A., 2006.
- [77] I. Millington. *Artificial Intelligence for Games*. Morgan Kaufmann Publishers, 2006.
- [78] S. Musse, F. Garat, and D. Thalmann. Guiding and interacting with virtual crowds in real-time. In *Proceedings of the Eurographics Workshop on Computer Animation '99, CAS '99*, pages 23–34, Milan, Italy, 1999.
- [79] S. Musse and D. Thalmann. A model of human crowd behavior: Group inter-relationship and collision detection analysis. In *Proceedings of Computer Animation and Simulation 1997, CAS '97*, pages 39–51, Budapest, Hungary, 1997.
- [80] S. Musse and D. Thalmann. Hierarchical model for real time simulation of virtual human crowds. In *IEEE Transactions on Visualization and Computer Graphics*, volume 7, pages 152–164, 2001.
- [81] N. Noser, O. Renault, D. Thalmann, and N. Thalmann. Navigation of digital actors based on synthetic vision, memory, and learning. *Computer and Graphics*, 19:7–19, 1995.
- [82] R. Parent. *Computer Animation: Algorithms and Techniques*. Morgan Kaufmann, 2001.
- [83] J. Pauls. Calculating evacuation times for tall buildings. *Fire Safety Journal*, 12:213–236, 1987.

- [84] N. Pelechano, J. Allbeck, and N. Badler. Controlling individual agents in high-density crowd simulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation '07, SCA '07*, San Diego, U.S.A., 2007.
- [85] J. Pettr, P. De Heros Ciechomski, J. Main, B. Yersin, P. Laumond, and D. Thalmann. Real-time navigating crowds: Scalable simulation and rendering. *Computer Animation and Virtual Worlds*, 17:445–455, 2006.
- [86] G. Pezzulo, E. Lorini, and G. Calvi. How do i know how much i don't know? a cognitive approach about uncertainty and ignorance. In *Proceedings of the 26th Annual Meeting of the Cognitive Science Society, CogSci '04*, Chicago, U.S.A, 2004.
- [87] M. I. Posner and S. E. Petersen. The attention system of the human brain. *Annual Review of Neuroscience*, 13:25–42, 1990.
- [88] M. I. Posner, C. R. R. Snyder, and D. J. Davidson. Attention and the detection of signals. *Journal of Experimental Psychology: General*, 109:160–174, 1980.
- [89] D. Pottinger. Terrain analysis in real-time strategy games. In *Proceedings of the Game Developer Conference '00, GDC '00*, 2000. <http://www.gamasutra.com/features/gdcarchive/2000/pottinger.doc>.
- [90] I. Prigonine and R. Herman. *Kinetic Theory of Vehicular Traffic*. American Elsevier, 1971.
- [91] A. S. Rao and M. P. Georegff. Modelling rational agents within a bdi architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484, 1999.
- [92] C. W. Reynolds. Steering behaviours for autonomous characters. In *Proceedings of Game Developer Conference*, pages 763–782, San Francisco, CA, USA, 1999.

- [93] S. J. Rymill and N. A. Dodgson. A psychologically based simulation of human behaviour. *EG UL Theory and Practice of Computer Graphics*, pages 25–42, 2005.
- [94] N. Saiwaki, T. Komatsu, T. Yohsida, and S. Nishida. Automatic generation of moving crowd using chaos model. *Systems, Man, and Cybernetics*, 4:3715–3721, 1997.
- [95] T. Sakuma, T. Mukai, and S. Kuriyama. Psychological model for animating crowded pedestrians. *Computer Animation and Virtual Worlds*, 16:343–351, 2005.
- [96] H. Samet. The quadtree and related hierarchical data structure. *ACM Computing Surveys*, 16:187–260, 1984.
- [97] A. Schadschneider, A. Kirchner, and K. Nishinari. Ca approach to collective phenomena in pedestrian dynamics. In *Proceedings of 5th International Conference on Cellular Automata for Research and Industry*, pages 239–248, Geneva, Switzerland, 2002.
- [98] W. Shao and D. Terzopoulos. Autonomous pedestrians. In *Proceedings of the 2005 ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 19–28, 2005.
- [99] T. S. Shen. Ems: A building evacuation simulation model. *Building and Environment*, 40:671–680, 2005.
- [100] T. J. Shields and K. E. Boyce. A study of evacuation from large retail stores. *Fire Safety Journal*, 35:25–49, 2000.
- [101] K. Shriver. Influences of environmental design on pedestrian travel behaviour in four austin neighbourhoods. *Journal Transportation Research Record*, 1578:64–75, 1997.
- [102] M. Smithson. *Ignorance and Uncertainty: Emerging Paradigms*. Springer, 1989.

- [103] G. Snook. chapter Simplified 3D Movement and Path-Finding Using Navigation Meshes. Charles River Media, 2000.
- [104] A. Stentz. The focussed d* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence '95*, pages 1652–1659, 1995.
- [105] C. Stephenson. Fast fits: New methods for dynamic storage allocation. In *Proceedings of the 9th Symposium on Operating Systems Principles*, pages 30–32, New Hampshire, U.S.A, 1983.
- [106] W. Sterren. *Path Look-up Tables- Small Is Beautiful*, pages 115–131. Charles River Media, 2004.
- [107] G. Still. *Use of Crowd Dynamics Around the World*. PhD thesis, Robert Gordon’s Institute of Technology, 2000.
- [108] N. Sturtevant and M. Buro. Partial pathfinding using map abstraction and refinement. In *Proceedings of the 20th National Conference on Artificial Intelligence, AAAI '05*, pages 1392–1398, Pittsburgh, U.S.A, 2005.
- [109] S. Surasmith. *AI Game Programming Wisdom*, chapter Preprocessed Solution for Open Terrain Navigation, pages 161–170. Charles River Media, 2002.
- [110] P. Sweetser. chapter Strategic Decision Making with Neural Networks and Influence Maps, pages 439–446. Charles River Media, 2004.
- [111] A. Szarowicz et al. The application of ai to automatically generate animation. *AI 2001: Advances in Artificial Intelligence*, 2256:393–416, 2001.
- [112] K. Teknomo. Applications of microscopic pedestrian simulation model. *Transportation Research Part F: Traffic Psychology and Behaviour*, 9:15–27, 2006.
- [113] K. Teknomo, Y. Takeyama, and H. Inamura. Microscopic pedestrian simulation model to evaluate *Lane-Like Segregation* of pedestrain crossing. In

- Proceedings of Infrastructure Planning Conference 2001*, volume 24, pages 137–140, Kouchi, Japan, 2001.
- [114] D. Thalmann. A new generation of synthetic actors: The interactive perceptive actors. In *Proceedings of Pacific Graphics 1996*, pages 200–219, Taipei, Taiwan, 1996.
- [115] D. Thalmann et al. Virtual humans’ behaviour: Individual groups and crowds. In *Proceedings of Digital Media Futures*, Bradford, UK, 1999.
- [116] C. Thureau, C. Bauckhage, and G. Sagerer. Learning human-like movement behaviour for computer games. In *Proceedings of 8th International Conference on Simulation of Adaptive Behaviour, SAB ’04*, pages 315–322, Los Angeles, U.S.A, 2004.
- [117] P. Tozour. *Game Programming Gems 2*, chapter Influence Mapping, pages 287–298. Charles River Media, 2002.
- [118] P. Tozour. *AI Game Programming Wisdom 2*, chapter Search Space Representation, pages 85–102. Charles River Media, 2004.
- [119] A. Treuille, S. Cooper, and Z. Popovic. Continuum crowds. In *Proceedings of SIGGRAPH ’06*, pages 1160–1168, 2006.
- [120] B. Ulicny and D. Thalmann. Towards interactive real-time crowd behaviour. *Computer Graphics Forum*, 21:767–775, 2002.
- [121] L. N. Vaserstien. *Introduction to Linear Programming*. Prentice Hall, 2003.
- [122] M. Weber. *The Theory of Social and Economic Organization*. Free Press, 1947.
- [123] S. White and C. Christensen. A fast approach to navigation meshes. *Game Programming Gems 3*, 2002.
- [124] P. Wilson, S. Mark, and J. Michael. Dynamic storage allocation: A survey and critical review. pages 1–117, Kinross, U.K., 1995.

- [125] S. Wolfram. Cellular automata as models of complexity. *Nature*, 311:419–424, 1984.
- [126] L. Z. Yang et al. Simulation of the kin behaviour in building occupant evacuation based on cellular automata. *Building and Environment*, 40:411–415, 2005.
- [127] W. K. Yeap and J. M. E. Computing a representation of the local environment. *Artificial Intelligence*, 107:265–301, 1999.
- [128] W. Yu, R. Chen, L. Dong, and D. S.Q. Centrifugal force model for pedestrain dynamics. *Phyiscal Review E*, 72:1–7, 2005.
- [129] C. Zimring, A. Joseph, G. Nicoll, and S. Tsepas. Influences of building design and site design on physical activity. *American Journal of Preventive Medicine*, 28:186–193, 2005.

Appendix A Proof of the Collision Speed Constraint

The agent's speed constraint can be used to describe the agent's maximum attainable speed or the agent's urgency to reach a destination fast. In this section, the mathematical proof for the agent's speed constraint will be presented. Corollaries A-1 to A-3 are a direct result of Theorem A-1. They are the specific constraints that are used to drive the agent's speed constraint based on the actual value of k_3 , i.e., they are the actual constraints that has been combined with the agent's collision avoidance constraint to generate the new steering parameters for the source agent to both avoid a target agent and observe the speed constraints of the agent. Note that the speed constraint in this form is necessary only if the speed constraint needs to be combined with the agent's collision avoidance constraint.

Theorem A-1: Let l_2 be the constraint on $\theta_s(t + \Delta t)$ to enforce the condition $s_{min} \leq s_s(t + \Delta t) \leq s_{max}$ as stated in Equation 4.17, when $\sin(\theta_g(t) - \theta_r(t + \Delta t)) \neq 0$. Then l_2 is given by:

If $\sin(\theta_g - \theta_r) < 0$,

$$\frac{k_3}{s_{min}} \leq \sin(\omega_s - k_0) \leq \frac{k_3}{s_{max}} \quad (\text{A.1})$$

else if $\sin(\theta_g - \theta_r) > 0$,

$$\frac{k_3}{s_{max}} \leq \sin(\omega_s - k_0) \leq \frac{k_3}{s_{min}} \quad (\text{A.2})$$

where $\omega_s(t + \Delta t)$ is the angular distance to travel by the source agent to avoid a potential collision and $k_3 = s_g(t)(t + \Delta t) \sin(\theta_g(t) - \theta_r(t + \Delta t))$.

Proof:

Using Equation 4.8 and 4.17

$$s_{min} \leq \frac{\sin(\omega_s - k_0)}{s_g \sin(\theta_g - \theta_r)} \leq s_{max}$$

When $\sin(\theta_g - \theta_r) > 0$

$$\frac{s_{min}}{s_g \sin(\theta_g - \theta_r)} \leq \sin(\omega_s - k_0) \leq \frac{s_{max}}{s_g \sin(\theta_g - \theta_r)}$$

When $\sin(\theta_g - \theta_r) < 0$

$$\frac{s_{min}}{s_g \sin(\theta_g - \theta_r)} \geq \sin(\omega_s - k_0) \geq \frac{s_{max}}{s_g \sin(\theta_g - \theta_r)}$$

Theorem A-1 expresses the speed constraint of the source agent as a sinusoidal function bounded by two horizontal lines. The next few corollaries will attempt to establish the speed constraint to be in terms of $\omega_s(t + \Delta t)$ based on different values of the two horizontal bounds.

Corollary A-1: Let $a \leq \sin(\omega_s(t + \Delta t) - k_0) \leq b$ be the general form of Theorem 9, then if $a \geq 0$ and $b \geq 0$, the solution set of $\omega_s(t + \Delta t)$ is defined by l_{21} such that

$$\begin{aligned} (\alpha + k_0 \leq \omega_s \leq \beta + k_0) \quad \vee \\ (\pi - \beta + k_0 \leq \omega_s \leq \pi - \alpha + k_0) \end{aligned}$$

where $\alpha = \arcsin(a)$, $\beta = \arcsin(b)$ and $k_0 = \theta_r(t + \Delta t) - \theta_s(t)$.

Corollary A-2: Let $a \leq \sin(\omega_s(t + \Delta t) - k_0) \leq b$ be the general form of Theorem 9, then if $a \leq 0$ and $b \geq 0$, the solution set of $\omega_s(t + \Delta t)$ is defined by l_{32} , such that

$$\begin{aligned} (-\pi + k_0 \leq \omega_s \leq -\pi - \alpha + k_0) \quad \vee \\ (-\alpha + k_0 \leq \omega_s < \beta + k_0) \quad \vee \\ (\pi - \beta + k_0 \leq \omega_s - k_0 \leq \pi + k_0) \end{aligned}$$

where $\alpha = \arcsin(a)$, $\beta = \arcsin(b)$ and $k_0 = \theta_r(t + \Delta t) - \theta_s(t)$.

Corollary A-3: Let $a \leq \sin(\omega_s(t + \Delta t) - k_0) \leq b$ be the general form of Theorem 9, then if $a \leq 0$ and $b \geq 0$, the solution set of $\omega_s(t + \Delta t)$ is defined by l_{33} , such that

$$\begin{aligned} (\beta - \pi + k_0 \leq \omega_s \leq \alpha - \pi + k_0) \quad \vee \\ (\alpha + k_0 \leq \theta_s \leq \beta + k_0) \end{aligned}$$

where $\alpha = \arcsin(a)$, $\beta = \arcsin(b)$ and $k_0 = \theta_r(t + \Delta t) - \theta_s(t)$.

Corollaries A-1 to A-3 states the speed constrain of the source agent in terms of $\omega_s(t + \Delta t)$. Figure A.1 explains how the corollaries are derived. The corollaries can be obtained directly by analysing a *sin* graph and how horizontal lines $y = a$ and $y = b$ cuts the *sin* graph at different points.

Figure A.1: Speed constraints in terms of ω_s .

The shaded regions represent valid values of $\omega_s - k_0$ that correspondingly describes the agent's speed constraints. One of these set of corollaries can then be combined together with the rest of the agent's collision avoidance constraint to take into consideration the agent's speed constraints during collision avoidance.

Appendix B Proof of the Multi-agent Avoidance Constraint

The ability of the agent to avoid multiple target agents in a dense pedestrian crowd is necessary in order to prevent the deadlocks as we have described in Section 6.1.1. In this section we will derive a relationship between two relative frames, R_{sg1} and R_{sg2} , and how the changes in one relative frame will affect the other.

We begin our derivation by looking at Figure B.1 which shows a combined vector diagram between two relative frames of a source agent. The first relative frame represents the sensory memory between of the source agent about the target agent. This relative frame is defined by $\vec{v}_{r1}(t + \Delta t)$, $\vec{v}_s(t)$ and $\vec{v}_{g1}(t)$. The second relative frame describes the sensory memory of the source agent about a neighbouring agent. This relative frame is defined by $\vec{v}_{r2}$, $\vec{v}_s(t)$ and $\vec{v}_{g2}(t)$. In general, $\vec{v}_{gi}(t)$ is the velocity of target agent i and $\theta_{ri}(t + \Delta t)$ is the orientation of the relative velocity between the source agent and target agent i . $\theta_s(t)$ and $\vec{v}_s(t)$ is the orientation and velocity of the source agent respectively.

From Figure B.1, we can observe how the changes in the relative frame between the source and target agent, a_{g1} , can affect the relative frame between the source and the second target agent, a_{g2} . In particular, we can observed that the relative orientation, $\theta_{r2}(t)$, between the source agent, a_s , and a_{g2} changes as the relative speed, $s_r(t)$, between the a_s and a_{g1} changes. This is a very interesting property because collision is avoided by ensuring that Constraint 4.7 is satisfied, i.e., that $\theta_{ri}(t + \Delta t)$ lies outside the collision

Figure B.1: Combined vector diagram of the two relative frames.

zone between the source agent and the target agent i . The same argument can be used to ensure that the source agent does not collide with the other target agents while avoiding the target agent, by ensuring the relative velocity between the source and other target agents do not lie within the collision zone of the respective part of agents.

Let $\theta_{r2 \min}$ and $\theta_{r2 \max}$ be the limit of the collision zone between the a_s and a_{g2} . Then by finding the value of $s_{r \min}$ and $s_{r \max}$ for the $\theta_{r2 \min}$ and $\theta_{r2 \max}$, we will be able to find the constraints on the orientation of the source agent. Our first step is therefore to find a relation between θ_{r2} and s_r using the vector diagram shown in Figure B.1.

Theorem B-1: If a source agent is found to be on a collision course with a target agent then the relative speed between the source and target agent, $s_r(t + \Delta t)$ when expressed in terms of the velocities of a neighbouring agent is given by

$$s_r = \frac{s_{g2} \sin(\theta_{g2} - \theta_{r2}) - s_{g1} \sin(\theta_{g1} - \theta_{r2})}{\sin(\theta_{r1} - \theta_{r2})} \quad (\text{B.1})$$

where $s_{g1}(t)$ and $\theta_{g1}(t)$ is the speed and orientation of the target agent, $s_{g2}(t)$ and $\theta_{g2}(t)$ is the speed and orientation of the target agent, a_{g2} . $\theta_{r1}(t + \Delta t)$ and $\theta_{r2}(t + \Delta t)$ is the euler angle of the relative velocities between the source and target agent and between the source and second target agent respectively.

Proof:

Let s_{r2} be the relative speed between the source agent and the second target agent,

$$\vec{v}_{g2} + \vec{v}_{r2} = \vec{v}_{g1} + \vec{v}_{r1}$$

$$s_{g1} \sin \theta_{g1} + s_r \sin \theta_{r1} = s_{r2} \sin \theta_{r2} + s_{g2} \sin \theta_{g2} \quad (\text{B.2})$$

$$s_{g1} \cos \theta_{g1} + s_r \cos \theta_{r1} = s_{r2} \cos \theta_{r2} + s_{g2} \cos \theta_{g2} \quad (\text{B.3})$$

Equation B.2/ Equation B.3:

$$s_{r2} = \frac{s_{g1} \sin \theta_{g1} + s_r \sin \theta_{r1} - s_{g2} \sin \theta_{g2}}{\sin \theta_{r2}} \quad (\text{B.4})$$

Substitute Equation B.4 into Equation B.3:

$$s_{g1} \sin(\theta_{g1} - \theta_{r2}) + s_r \sin(\theta_{r1} - \theta_{r2}) = s_{g2} \sin(\theta_{g2} - \theta_{r2})$$

$$\therefore s_r = \frac{s_{g2} \sin(\theta_{g2} - \theta_{r2}) - s_{g1} \sin(\theta_{g1} - \theta_{r2})}{\sin(\theta_{r1} - \theta_{r2})} \quad (\text{B.5})$$

Theorem B-1 therefore states the relationship between the $s_r(t)$ and $\theta_{r2}(t)$ so that a specific $s_r(t)$ can be found for each value of $\theta_{r2}(t)$. In particular we need two values of $\theta_{r2}(t)$ to define the collision zone between the source and second target agent. Let these two limits on $\theta_{r2}(t)$ be defined as $\theta_{r2 \min}$ and $\theta_{r2 \max}$ such that

$$\theta_{r2 \min} < \theta_{r2} < \theta_{r2 \max} \quad (\text{B.6})$$

represents the collision zone between the source agent and the second target agent. Each value of $\theta_{r2}(t)$ then will yield a value of $s_r(t)$ using Equation B.1. Let the two values of $s_r(t)$ be $s_{r \min}$ and $s_{r \max}$ such that

$$s_{r \min} \leq s_{r \max} \quad (\text{B.7})$$

Then the next step will be to find the values of $\theta_s(t + \Delta t)$ that corresponds to $s_{r \min}$ and $s_{r \max}$. In particular the range of $\theta_s(t + \Delta t)$ will describe the collision zone between the source and second target agent. Therefore, in order to avoid both the first and second target agents together, we merely have to exclude the values of $\theta_s(t + \Delta t)$ that describes the collision zone of both the first and second target agents.

Theorem B-2: Given the value of $s_r(t + \Delta t)$ between the source and target agent, then the value of $\tan \theta_s(t + \Delta t)$ is given by

$$\tan \theta_s = \frac{s_{g1} \sin \theta_{g1} + s_r \sin \theta_{r1}}{s_{g1} \cos \theta_{g1} + s_r \cos \theta_{r1}} \quad (\text{B.8})$$

where $s_{g1}(t)$ and $\theta_{g1}(t)$ is the speed and orientation of the target agent. $s_r(t + \Delta t)$ and $\theta_{r1}(t + \Delta t)$ is the speed and orientation of the relative velocity between the source and target agent.

Proof:

$$\vec{v}_{g1} + \vec{v}_{r1} = \vec{v}_s$$

$$s_{g1} \sin \theta_{g1} + s_r \sin \theta_{r1} = s_s \sin \theta_s \quad (\text{B.9})$$

$$s_{g1} \cos \theta_{g1} + s_r \cos \theta_{r1} = s_s \cos \theta_s \quad (\text{B.10})$$

Equation B.9 / B.10:

$$\tan \theta_s = \frac{s_{g1} \sin \theta_{g1} + s_r \sin \theta_{r1}}{s_{g1} \cos \theta_{g1} + s_r \cos \theta_{r1}} \quad (\text{B.11})$$

Theorem B-2 therefore allows the value of $\theta_s(t + \Delta t)$ to be found based on the values of $s_{r \min}$ and $s_{r \max}$. Similar to Theorem B-1, there will be two values of $\theta_s(t + \Delta t)$, one corresponding for each corresponding value of s_r . We define $\theta_{s \min}$ and $\theta_{s \max}$ to represent the two values of $\theta_s(t + \Delta t)$ such that

$$\theta_{s \min} \leq \theta_{s \max} \quad (\text{B.12})$$

The normalized constraint, l_4 on $\theta_s(t + \Delta t)$ for each neighboring agent in the environment that the source agent is aware about is hence given by Equation B.13.

$$l_4 = 0 \leq \theta_s < \theta_{s \min} \cup \theta_{s \max} < \theta_s \leq 2\pi \quad (\text{B.13})$$

Constraint l_4 can then be combined with the rest of the constraints discussed in Section 4.1 using the method described in Section 4.1.5. Each of this constraint will ensure that the source agent will not collide with a neighbouring agent while avoiding the target agent. This means that for each neighbouring agent that the source agent is aware of, one l_4 is required for the agent to avoid plotting a collision course with that neighbouring agent. For example, a source agent who is aware of five target agents will need one set of collision avoidance constraints $l_1 \cup l_2$ and four l_4 constraints to avoid all five target agents simultaneously.

Appendix C Author's Publications

During the period working on this thesis, the author has published several refereed papers in international conferences and journals. These papers are listed as follows:

1. W.L. Koh and S.P. Zhou. An extensible collision avoidance model for realistic self-driven autonomous agents. *In Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pages 7-14, Crete, Greece, 2007. **Nominated for best paper award.**
2. W.L. Koh, L. Lin and S.P. Zhou. Modelling and simulation of pedestrian behaviours. *In Proceedings of the 22nd Workshop on Principles of Advanced and Distributed Simulation*, pages 43-50, Rome, Italy, 2008. Acceptance Rate: 35%.
3. W.L. Koh and S.P. Zhou. A novel micro-level navigation model for simulating realistic pedestrian behaviours. *IEEE Transactions on Systems, Man, and Cybernetics Part B*. (**submitted**)
4. W.L. Koh and S.P. Zhou. Individuals in a crowd: modelling the mind of a pedestrian. *ACM Transactions on Modelling and Computer Simulation*. (**submitted**)