MODELING REALISTIC HIGH DENSITY AUTONOMOUS AGENT CROWD MOVEMENT: SOCIAL FORCES, COMMUNICATION, ROLES AND PSYCHOLOGICAL INFLUENCES

NURIA PELECHANO GOMEZ

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2006

__________________________

Dr. NORMAN I. BADLER
Supervisor of Dissertation

__________________________

Dr. RAJEEV ALUR
Graduate Group Chairperson
COPYRIGHT
Nuria Pelechano Gómez
2006
To my parents, my sister and my fiancé
ACKNOWLEDGMENTS

This dissertation has benefited from the comments and suggestions of many people and thus I would like to take this opportunity to express my gratitude.

Firstly, I would like to thank my advisor, Professor Norman Badler, for his helpful guidance and support over the past three years. I feel very fortunate to have had the experience of doing research with him.

While conducting this research, I had the opportunity to work with and learn from many people. I am especially grateful to Jan Allbeck for her thoughtful advice and encouragement. I am also thankful to all the students at the Center for Human Modeling and Simulation for their feedback and advice during our reading group sessions. I would also like to thank Professor Ali Malkawi for giving me the opportunity to participate in some evacuation simulation projects for the Building and Simulation Group at the University of Pennsylvania.

I also gratefully acknowledge professors Vijay Kumar, Stephen Lane, Mark Yim and Craig Reynolds for serving on my dissertation defense committee.

Financial support for this research was provided in part by a Fulbright fellowship, NASA and Army UOFACT.

I thank my family for supporting me during all the years I have been far away from home, especially my parents and my sister.

Finally, I thank my fiancée, Peter Williams, for his love, patience, advice, encouragement and time dedicated to review my work. For many reasons, this dissertation would have not been possible without him.
ABSTRACT
MODELING REALISTIC HIGH DENSITY AUTONOMOUS AGENT CROWD MOVEMENT: SOCIAL FORCES, COMMUNICATION, ROLES AND PSYCHOLOGICAL INFLUENCES

Nuria Pelechano
Supervisor: Norman Badler

The simulation of realistic, large, dense crowds of autonomous agents is still a challenge for the computer graphics community. Typical approaches either look like particle simulations (where agents ‘vibrate’ back and forth) or are conservative in the range of motion possible (agents aren’t allowed to ‘push’ each other). Our HiDAC system (High Density Autonomous Crowds) focuses on the problem of simulating the local motion behaviors of crowds moving in a natural manner within dynamically changing virtual environments. This is achieved by applying a combination of psychological and geometrical rules layered on top of a social forces model. The results show: elimination of agent ‘shaking’ behavior, fast perception, and a wide variety of emergent behaviors including: bi-directional flows, overtaking, emergent queuing with different line widths, agents being ‘pushed’ and ‘falling’, and panic propagation. These behaviors emerge based on the current situation, agent personality and perceived density of the crowd.

To accurately simulate crowds in large, complex environments, it is not enough to only model local motion; agents must also have the ability to navigate the unknown virtual environment. We therefore address the problems that arise during crowd navigation where not all individuals have complete knowledge of the building’s internal structure. In addition, we simulate the effects of communication on the behavior of autonomous agents while exploring the building. We have developed a system called MACES (Multi-Agent Communication for Evacuation Simulation) which combines local motion with wayfinding using inter-agent communication and
different roles. Together they automatically augment an agent’s mental map of the environment to produce empirically better maze evacuation performance.

We study the emergent behavior during building evacuation under different conditions such as agents using communication to share their knowledge of the building routes and hazards, psychological factors driving different navigation skills, and agents taking different roles such as trained personnel, leaders and followers. The experimental results show significant improvements in evacuation rates with inter-agent communication and demonstrate that only a relatively small percentage of trained leaders yield evacuation rates comparable to the case where all are trained.

The framework presented in this dissertation combines decision making, including communication and roles (MACES), with local motion (HiDAC). The two systems interact in real-time while being driven by a set of psychological and physiological parameters that allow the user to have control over the final behavior exhibited by the crowd.
2.6.2. Flow tiles and potential field methods ...................................................... 40
2.6.3. Probabilistic Roadmaps (PRMs) ............................................................... 40
2.7. Environment Modeling ............................................................................. 41
3 Building Modeling ......................................................................................... 43
3.1. Cell and Portal graph automatic generation .............................................. 45
3.1.1. Generate cell and portal graph for each floor ........................................... 45
3.1.2. Identify stairs and link floors through new cells ................................. 48
3.1.3. Identify and store walls ........................................................................ 49
3.1.4. Identify and store obstacles ................................................................. 50
3.2. Pre-calculating data for real-time simulation ........................................... 51
4 Framework .................................................................................................... 55
4.1. Interaction between high level, low level and psychological models .......... 58
4.2. Parameters affecting crowd behavior ...................................................... 61
5 Low level: Local Motion ................................................................................. 66
5.1. Introduction ............................................................................................... 67
5.2. Agents’ speeds and densities ................................................................. 68
5.2.1. Walking speeds and densities when walking downstairs ....................... 72
5.3. Perception .................................................................................................. 73
5.4. Basic Social Forces model ...................................................................... 74
5.5. HiDAC system .......................................................................................... 78
5.5.1. Extending the social forces model ......................................................... 78
5.5.2. Solution for the “shaking” problem in high-density crowds ................. 88
5.5.3. Overtaking and bi-directional flow behavior ........................................ 90
      Animation Results on Bi-directional flows and overtaking animation ....... 94
5.5.4. Organized behavior - queuing ............................................................... 98
      Animation Results on queuing behavior .................................................. 100
5.5.5. Pushing behavior ............................................................................... 101
      Animation Results on pushing behavior ................................................ 103
5.5.6. Falling and becoming obstacles ......................................................... 104
Animation results on falling and becoming new obstacles .................. 106

5.5.7. Panic Propagation ........................................................................... 107

6 Wayfinding with Communication and Roles ........................................ 109

6.1. Introduction ............................................................................................. 109

6.2. High level algorithm .............................................................................. 110

6.2.1. Exploring the building ....................................................................... 112

6.3. Results on Wayfinding and Decision-Making ...................................... 114

6.3.1. Comparison of random search vs. Depth First Search ................. 115

6.3.2. Comparison of communication vs. non communication ............... 116

6.3.3. Comparison of trained leaders vs. untrained leaders .................... 118

6.3.4. Importance of leadership ................................................................. 120

6.3.5. Psychology affecting roles and navigation ....................................... 122

6.3.6. Interactively modifying navigation and impatient agents avoiding

bottlenecks .................................................................................................... 126

7 Summary of Results .................................................................................. 130

7.1. Performance ........................................................................................... 133

8 Conclusions ................................................................................................ 136

Limitations and Future Work Plan .............................................................. 140

9 References .................................................................................................. 142

APPENDIX A: Building Editor ................................................................. 153
List of Tables

Table 1: Definition of a cellular automaton ................................................................. 18
Table 2: Comparison of different systems for animation of large groups .............. 34
Table 3: Algorithm to assign cell IDs to free space .................................................. 46
Table 4: Algorithm to propagate IDs ........................................................................ 46
Table 5: Algorithm for shortest paths propagation ................................................... 53
Table 6: Alternative paths example .......................................................................... 54
Table 7: Parameters affecting behavior ...................................................................... 61
Table 8: Velocity factors ............................................................................................ 69
Table 9: Mean velocities ............................................................................................ 69
Table 10: Fruin levels of Service ............................................................................... 70
Table 11: Summary of walking speeds according to several studies [TM95a] ......... 71
Table 12: Maximum and Ultimate flow rates [TM95a] .............................................. 71
Table 13: Objects classification and repulsive priorities .......................................... 80
Table 14: Procedure to calculate the next position .................................................... 87
Table 15: Braking forces algorithm .......................................................................... 89
Table 16: Panic values and the corresponding waiting radius .................................. 98
Table 17: High level results summary ....................................................................... 131
Table 18: Goals and results for low level motion ...................................................... 132
List of Figures

Figure 1: Current models framework and our approach for low level motion, HiDAC ................................................................................................................................. 11
Figure 2: 3D environment and its corresponding grid of cells. ............................ 16
Figure 3: Reynolds’ boids..................................................................................... 20
Figure 4: Only one person at a time can fit through the opening of the stairwell, when the actual width is 1m. (STEPS)................................................................. 36
Figure 5: Paths followed based on potential maps in Exodus ............................. 37
Figure 6: Uneven utilization of stairwells during route selection (STEPS)........... 38
Figure 7: Example of a maze used for our experiments with 2 exits and 8 hazards .. 43
Figure 8: Example of building plan used for evacuation simulations. ................ 44
Figure 9: Building geometry (a) and grid partition with walls, obstacles and door (b) ............................................................................................................................ 47
Figure 10: Eight first steps of the cell identification and ID propagation algorithm.. 47
Figure 11: Cell and Portal graph......................................................................... 48
Figure 12: Cell and Portal graph with stairwells ................................................. 49
Figure 13: Walls assigned to each room and door attractors............................ 50
Figure 14: Shortest paths propagation and distances......................................... 52
Figure 15: Example of a complex building ......................................................... 56
Figure 16: Architecture overview ................................................................. 59
Figure 17: Architecture overview with psychological model............................ 60
Figure 18: MACES+HiDAC interface ............................................................... 62
Figure 19: Relationship between speed and density walking downstairs during evacuation [SFPE02] ................................................................................. 72
Figure 20: Relationship between flow and density walking downstairs [SFPE02].... 72
Figure 21: Perception for the yellow agent....................................................... 74
Figure 22: Arching in the doorways. ............................................................... 75
Figure 23: Agents steered by attraction points ........................................................... 76
Figure 24: Crossing Portals States .............................................................................. 77
Figure 25: Agents crossing a portal ............................................................................ 77
Figure 26: Personal space of an agent. Showing a case where one is compromised but not the other. ....................................................................................................... 81
Figure 27: Repulsion force with the walls ..................................................................... 83
Figure 28: Avoidance forces with agents and round obstacles ....................................... 84
Figure 29: Avoidance forces for walls .......................................................................... 85
Figure 30: Example of repulsion forces which are necessary to apply braking forces. .................................................................................................................................................. 88
Figure 31: Collision Avoidance rectangle of influence ..................................................... 91
Figure 32: Bi-directional flows ..................................................................................... 93
Figure 33: Results on bi-directional flows for low densities ............................................. 94
Figure 34: High density crowds bi-directional flows based on right preference .......... 96
Figure 35: Example of overtaking animation .................................................................. 97
Figure 36: Areas of influence for waiting behaviors ....................................................... 99
Figure 37: Examples of wide and thin queues emerging when animating a “normal” scenario ......................................................................................................................... 99
Figure 38: Different emergent queuing behavior based on the weights of the tangential forces. Weights from top-left to bottom-right are 0.1, 0.5, 2.0, 5.0. ........................................ 100
Figure 39: Pushing forces ........................................................................................... 101
Figure 40: Red-headed people exhibit panic behavior and push others to open their way through the crowd ..................................................................................................... 103
Figure 41: Results of people being pushed off course by the crowd .............................. 104
Figure 42: Agents avoiding a body on the floor .............................................................. 105
Figure 43: Avoidance forces around fallen agents ....................................................... 105
Figure 44: Agents avoiding fallen individuals .............................................................. 106
Figure 45: Agents walking over fallen individuals ....................................................... 107
Figure 46: Panic propagation sequence ....................................................................... 108
Figure 47: Example of complex building ................................................................. 110
Figure 48: High Level Wayfinding diagram ............................................................. 111
Figure 49: Exploration diagram ........................................................................... 113
Figure 50: Communication vs. non communication ............................................. 116
Figure 51: Evacuation time for different crowd sizes using communication. 100% untrained leaders ......................................................................................... 117
Figure 52: Congestion at doors .......................................................................... 118
Figure 53: Evacuation time for 0, 25, 50, 75 and 100% leadership ..................... 118
Figure 54: Evacuation times for small percentages of trained leaders ............... 119
Figure 55: Sequence of crowd evacuation with high percentage of leadership ... 121
Figure 56: Sequence of crowd evacuation with low percentage of leadership .... 121
Figure 57: Autonomous agent exploring a virtual environment under normal conditions ................................................................. 123
Figure 58: Autonomous agent exploring a virtual environment in panic and therefore exhibiting disorientation ................................................................. 125
Figure 59: Results on interactive planning for impatient agents ......................... 127
Figure 60: Crowd reaction to dynamic changes in the environment .................... 129
Figure 61: Frame rate for 3D rendering vs. agent population ............................. 133
Figure 62: Milliseconds per frame vs. population size ........................................ 134
Figure 63: Frame rate vs. population size for simulation only ............................. 134
Figure 64: Milliseconds per Frame vs population size for simulation only ........ 135
Figure 65: Building Editor .................................................................................. 153
Figure 66: Example of high rise building editing and some 3D views ............... 154
Chapter 1

Introduction

There are many applications of computer animation and simulation where it is necessary to model virtual crowds of autonomous agents. Some of these applications include site planning, education, entertainment, training and human factor analysis for building evacuation. Other applications include simulations of scenarios where masses of people gather and then disperse, such as sporting events and concerts.

1.1. Motivation

Animating virtual crowds is often mediated by local rules [MT01], forces [HFV00], or flows [Che04]. The goal is usually either to achieve real time simulation for very large crowds where each individual’s behavior is not important as long as the overall crowd movement looks realistic, or to focus on individuals’ behavior using complex cognitive models (but achieving real time only for smaller crowds). Much effort has been put into improving the behavioral realism of each of these approaches, however none of the current models can realistically animate high density crowds. The next goal in crowd motion is therefore to be able to realistically animate high density crowds for real time applications, where agents are endowed with psychological elements that will drive not only their high level decision making, but also their reactive behavior (pushing, moving faster, being impatient, etc.).
On the global navigation level, the current approaches either deal with very simple environments or assume that agents have complete knowledge of the environment and move towards their goal as individuals (without interacting with other agents). Our motivation was therefore to focus on simulating realistically how communication affects the behavior of autonomous agents [CBBC99], and how combining different personalities and situations affect agents’ navigation and the way they interact with other agents and the environment [PB06].

It is therefore necessary to study the psychology literature in order to endow our agents with psychological factors that affect their overall behavior and movement as close as possible to what real humans exhibit. By using this information we expect to produce crowd animation that is “pre-validated”, by being based on known effects.

Crowd evacuation from large and complex public building spaces is usually hindered by a lack of knowledge of the detailed internal connectivity of the building’s rooms. In such circumstances, occupants may not be aware of the existence of suitable paths for circulation or, in the case of an emergency situation, the most appropriate paths of escape. This is a well known problem [Sim84]: building occupants usually decide to make use of familiar exits, which are often the way in which they entered the building. Emergency exits or exits not normally used for circulation are often ignored. When a fire situation occurs, where the smoke may also be obscuring vision and fire may be blocking some of those known paths, the problem is fatally aggravated.

Spatial awareness and orientation within buildings is made possible by the five senses. The perceived space depends on the individual ability and psychological condition of the person. Most people react to time pressure by an increase in the speed of their actions, as well as by subjectively choosing information. In general, the evacuation of a building due to imminent danger is accompanied by considerable physical and psychological stress. Since rising stress levels have the effect of diminishing the full functioning of one’s senses, this leads to a general reduction of awareness, especially the ability to orient oneself quickly in rooms and other surrounding areas [WSG03].
Decision skills in emergency situations are influenced by several factors such as the uncertainty of changes that might occur to the environment and the time pressure under which decisions have to be taken. If the agents have not been properly trained for these situations, they are likely to feel stressed and might reach the point where they find themselves incapable of making the right decision. “Stress occurs when there is a substantial imbalance between environmental demand and the response capability of the focal organism” [Gra70]. Time pressure is given by the difference between the amount of time available and the amount of required time to take a decision [RL93].

In contrast, trained individuals such as firefighters deal with a dynamically changing environment and choose the best sequence of actions based on their perception and knowledge of the environment. Their decision making process is biased and it is based on the importance they assign to each evaluative dimension, such as saving lives, keeping fire from spreading, minimizing risks for themselves or the rest of the team, etc. A specific decision could sometimes lead to failure, whereas the same decision at a different moment could be the best solution.

In an evacuation situation caused by a fire the main sources of stress could be too much or too little information coming at one time (several people in the same room taking different decisions and shouting different information about blocked rooms), complex and dynamically changing situations that result in uncertainty, and time pressure.

Sociologists and psychologists have studied the behavior of people in a crowd when they share a common goal. There has been much computational work done in crowd simulation using different methods to perform local motion such as cellular automata (CA), social forces models and rule-based systems. All these models simulate the movement of people within a familiar environment trying to achieve their goal while avoiding collisions with walls, obstacles and other individuals in the crowd.

Our work not only focuses on improving the group and individual motions for high density crowds within complex environments, but also we deal with a group of agents performing a high level wayfinding algorithm to obtain a cognitive map of a building. Wayfinding is the process of determining and following a route to some destination.
It deals with the cognitive component of navigation and therefore with the knowledge and the information processing required to get from an initial position to a goal position. Initially the individuals of the crowd would only have partial information about the internal building structure and as they explore it and communicate with other individuals in the crowd, they will be able to find paths towards the unblocked exits.

Wayfinding is defined as a spatial problem solving process with three subprocesses: decision making, decision execution, and information processing. In order to carry out wayfinding, each agent would need:

- a cognitive map: a mental model of the space
- an orientation: current position within the cognitive map
- exploration: processes to learn the features of the space (doors, walls, hazards, etc).
- navigation: the process of making its way through the environment

There have been several cognitive agent architectures proposed to generate human-like behavior. They generally consist of knowledge representation, algorithms that learn, and modules that plan actions based on that knowledge [FTT99, WCP_]. Tu and Terzopoulos have worked on behavioral animation for creating artificial life, where virtual agents are endowed with synthetic vision and perception of the environment [TT94, FTT99]. However, these systems are not easily scalable.

Rule-based schemes are fast enough for use with dozens of agents. Reynolds [Rey87] described the first use of a distributed behavioral model to produce flocking behavior. Brogan and Hodgins [BH97, BH02] used particle systems and dynamics for modeling the motion of groups with significant physics. Helbing [HFV00, HFMV02] describes methods to simulate the movement of pedestrians based on a social force model which is a microscopic (personal) approach for simulating pedestrian motion. This approach solves Newton’s equation for each individual and considers repulsive interactions, friction forces, dissipation and fluctuations.
These traditional crowd simulators ignore the differences between individuals and treat everyone as having the same simple behavior, but there are other models that represent each individual as being controlled by rules based on physical laws or behavioral models showing individualism [BMOB03]. In a multi-agent crowd system, the agents are autonomous, typically heterogeneous, and their concern is with coordinating intelligent behaviors among themselves, that is, how these agents can coordinate their knowledge, goals, skills, and plans to take action and to solve problems. Some of these applications include crowd behavioral models used in the training of military personnel [WSS01] and crowd motion simulations to support architectural design for both everyday use [BG96] and emergency evacuation conditions [Sti00, MT00, MT01].

Hierarchical schemes have been proposed to address scalability [FMS00]. In particular, Musse et. al. provides crowds with different levels of autonomy for hierarchical crowd behaviors [MT01], but complex individual behaviors have not been shown.

In multi-agent systems, each agent needs to sense the environment to perceive changes and react to them. Perception is often simulated by casting a set of rays and finding their intersection with obstacles around the object [ST05, PHL05]. Massive SW [MS05] (used, for example, in the ‘Lord of the Rings’ movies) has also developed a crowd simulation system with vision-based behavior. Individual agents do a low resolution render of the scene from their own point of view and use it to guide their actions.

In order to reduce the complexity of controlling all the agents in the crowd while detailed behaviors are still guaranteed, several systems have attached information to the environment [FBT99, TLCC01, TD00, PB06]. Our system also embeds information, such as shortest paths, in the environment. Individual agents will have differential types of access to that information and they will use it in different ways depending on their individual behavior at any given moment.
1.1.1. **Main features in crowd simulation systems**

There are a range of features that characterize crowd simulation systems. In this section we introduce a number of them and then provide a brief explanation of our current systems within this context.

a) *Crowd size*: refers to the number of individuals that the system can simulate in real time. Some applications need very large crowds to measure both the overall flow rate in different parts of the environment and percentage of crowd evacuated. Other systems focus on simulating realistic human behavior within a crowd. They usually work with smaller groups and study the interaction both between individuals and with the environment.

b) *Goal*: refers to whether individuals have one main task, such as walk towards one exit, or whether they have several routes to escape. In most of the simulations, the individuals are assumed to have complete knowledge of the environment. Systems that focus more on individual behaviors may also have some sub-goals within the simulation such as to help others to the exit or to do some specific action before walking towards the exit.

c) *Type of hazards*: some systems simulate drill evacuation and therefore there is no hazard. Others simulate only fire and the way it propagates, while the most complex ones also simulate smoke propagation, and the way it affects the individual’s performance depending on the level of toxicity. Other simulations do not occur within an evacuation scenario and have no hazards because their main focus is everyday scenarios such as people navigating in a train station during rush hour.

d) *Individuality/Role*: systems that are concerned with simulating realistic human behavior within a crowd implement microscopic approaches where
individuals have different decision making processes depending on their internal characteristics. These characteristics are generally given by a set of parameters whose values are assigned in a probabilistic manner.

e) **Communication/signaling/alarms:** deal with whether there is some kind of interaction between individuals and the environment. Some systems use alarms to start the evacuation process. Others implement simple signaling methods or instructions given by the firemen to indicate the evacuation routes.

f) **Behavior method:** the behavior of occupants is represented in many different ways and is explained in more detail in the Previous Work section (2). This feature can be implemented using a number of techniques such as: rule based models, physical models, Cellular Automata, and Finite State Machines.

g) **Spatial structure:** refers to whether the space is considered in a continuous way, or whether it is subdivided in some sort of grid (i.e. the space is divided into squares or hexagons)

h) **Hierarchical systems:** some systems implement different levels of behavior (scripted, autonomous). Also the crowd could be given as a multilayer architecture where behaviors are associated to individuals, groups, or the entire crowd. This allows for a wider variety of behaviors.

i) **Environment type:** such as home, train station, ship, multiple storey building, or outdoor environment.

In this work we have focused on achieving realistic motion for humans while they interact in real time with other agents and a dynamically changing environment. Our
agents’ local motion is computed by an extended social forces model, where psychological and geometrical rules are applied to modify the parameters affecting each agent’s movement. This results in a wide variety of emergent behaviors.

In our investigation of crowd wayfinding, we are dealing with groups of agents of different sizes (from 10 to 200) in a maze-like building. We have simulated the evacuation time taken by a group of agents to find the exits in a maze when an emergency occurs. The accident is assumed to happen simultaneously at several sites within the building. At that moment there will be different types of agents in the building. Some of them represent individuals that will not be very familiar with the environment, and therefore will just know a few paths within the maze towards the exits. Other agents are more familiar with the building and will have complete knowledge about alternative routes towards a destination room. Each agent has its own cognitive (or mental) map that is updated as the agent navigates the environment and communicates building path information with other agents.

1.2. Problem Statement

Animating motion for large crowds has been an important goal in the computer graphics, movie and video games communities. There has been a considerable amount of work on locomotion, path planning, navigation in large virtual environments and realistic behavior simulation using cognitive models.

In terms of defining the motion of each agent, we classify three main approaches: social forces systems, rule based models and cellular automata models. Much effort has been put into improving the behavioral realism of each of these approaches; none of the current models, however, can realistically animate high density crowds. Social forces models tend to give simulations that look closer to particle animation than human movement. Cellular automata models limit the movement of the agents, and tend to look like a checkerboard when the density is high. Finally, rule based models either do not consider collision detection and repulsion at all, or adopt very conservative approaches.
through the use of waiting rules. These rules work well for low densities in everyday life simulation, but lack realism for high density or panic situations.

The current models for crowd simulation in computer graphics cannot realistically handle body-to-body contact for large crowds and while they achieve good results for low and medium density crowds in normal situations (e.g. people walking in a train station, or virtual city), they fail to realistically simulate high density situations, such as an evacuation from a virtual building.

When simulating large groups of agents, it is not sufficient to only have realistic low level movement for the agents, it is also necessary to endow the agents with a high level behavior that can closely simulate the decision making process of real people. Most of the work in crowd simulation either deals with simple environments, or assumes every agent has complete knowledge of the environment. Therefore, it is necessary to simulate autonomous agents that can navigate unknown environments, learn their features and communicate with other individuals in the crowd in order to exchange relevant information to achieve their goals. Agents within a crowd should interact with other individuals as real people do, therefore communication is a very important feature in crowd simulation. In addition, agents must use perception to detect other agents’ positions to perform collision avoidance and to learn other agents’ psychological states and react to them. For example, an agent seeing another agent under panic and pushing through a crowd could trigger panic behavior which allows us to simulate panic propagation.

1.3. Contribution

This dissertation presents a framework to realistically simulate high density crowds in complex environments. Both the local motion and global wayfinding with communication and roles are driven by a set of parameters that define the psychological and physiological capabilities of the individual agents. To the extent that real-life data exists, these parameters are based on empirical models of actual crowd simulation.
These parameters can be tuned through an interface that allows us to visualize how changes in the personality of the agents are affected during a simulation of a given environment and situation. This multi-agent system is composed of two subsystems that interact to achieve the desired behaviors:

- Local motion: HiDAC (High Density Autonomous Crowds)
- Global navigation: MACES (Multi-Agent Communication for Evacuation Simulation)

HiDAC focuses on the problem of simulating high-density crowds of autonomous agents moving in a natural manner in dynamically changing virtual environments. We present a solution to the problem of realistically simulating local motion under different situations and agent personalities. Psychological and geometrical rules are layered on top of the basic social forces model in order to improve high-density crowd movement and realism. Since applying the same rules to all agents leads to homogeneous behavior, agents are given different psychological (e.g., impatience, panic) and physiological (e.g., speed) traits that trigger heterogeneous behaviors based on crowd density and personality. HiDAC exhibits a wide variety of emergent behaviors from agent line formation to pushing behavior and its consequences. These behaviors develop based on the current situation, personalities of the individuals and perceived density of the crowd.

Other emergent behaviors that our system exhibits include:

- Natural bi-directional distribution and flow rates
- Organized behavior – queuing
- Avoiding agents vibration
- Pushing through a crowd
- Agents falling and becoming obstacles
- Propagating panic
Figure 1 shows a taxonomy for crowd simulation and compares our model (HiDAC: High Density Autonomous Crowds) with the main models in the literature along the dimensions of animation realism and crowd density.

![Taxonomy diagram]

*Figure 1: Current models framework and our approach for low level motion, HiDAC*

MACES implements global wayfinding during building evacuation for crowds that may be unfamiliar with the internal connectivity of the environment. MACES combines local motion with wayfinding using inter-agent communication and different roles. Together they automatically augment an agent’s mental map of the environment to produce empirically better maze evacuation performance. We included individualism into the agents by assigning them different roles: trained leaders, untrained leaders, and followers. The flexibility of the model allows for variations in the number of people, building structure, number of hazards, and different role combinations for each agent. The main contribution of MACES is the development of a crowd simulation system that integrates:

(i) A global wayfinding algorithm to allow individuals in a crowd to explore an unfamiliar building in order to find exits during an emergency.

(ii) The use of inter-agent communication to share knowledge of the building during high level wayfinding.

(iii) The inclusion of agent roles to provide individualism and thus differences in their decision making processes.
(iv) Psychological elements affecting agent decision-making (e.g. panic causing disorientation and changes between roles, and impatience causing agents to dynamically replan their route)

MACES and HiDAC are integrated by passing relevant information between them. This involves the wayfinding module setting attractors that the low level module needs to perform local motion, and the low level motion querying the high level for new attractors when either intermediate goals are reached or changes are detected in the environment which require the high level to re-plan the route. Both systems (MACES and HiDAC) are supervised by a psychological and physiological module, which determines how the personality of the agent and the current state affects their overall performance in terms of navigation and local behavior.

The reminder of this thesis is organized as follows: first we will show some previous work in crowd animation and compare some of the academic and commercial software available against our system. Then we will describe how relevant navigation information is obtained from the geometry. Chapter 4 introduces the framework of the system we have developed that combines HiDAC and MACES. Chapter 5 explains HiDAC in depth showing detailed animation results. Chapter 6 explains MACES and presents the wayfinding results obtained. Finally, a summary of results and the conclusions of this dissertation are given.
Chapter 2

Previous Work

The focus of this section is to introduce the most relevant crowd simulation techniques and to present several systems that have been developed for animation or evacuation dynamics purposes.

The section starts with an introduction to pedestrian movement models and then addresses three parts. The first part represents microscopic methods, the second part presents macroscopic methods and the final part details several examples.

2.1. Microscopic and Macroscopic approaches used to model pedestrian movements

A large number of models for pedestrian simulation have been developed over the years in a variety of disciplines including computer graphics, robotics, and evacuation dynamics. These can be grouped into two main approaches: macroscopic and microscopic. Macroscopic models focus on the system as a whole, while microscopic models study the behavior and decisions of individual pedestrians and their interaction with other pedestrians in the crowd.

Microscopic models describe the space-time behavior of individual pedestrians. There are two subcategories: Social Force models and Cellular Automata (CA) models. The difference between them is in the discretization of space and time. Social Force models
[HFV00] describe pedestrian behavior microscopically by social fields (virtual “physical” forces) induced by the social behavior of the individuals. In the Cellular Automata approach the area under study is represented by a uniform grid of cells with local states depending on a set of rules, which describe the behavior of the pedestrians [Che04]. These rules compute the state of a particular cell as a function of its previous state and the states of the adjacent cells.

### 2.2. Microscopic Models

#### 2.2.1. Social force models

The social force model of pedestrian motion is a very advanced microscopic approach for simulating pedestrian motion. It solves Newton’s equation for each individual and considers repulsive interaction, friction forces, dissipation and fluctuations. This model can be successfully applied to simulate real world scenarios in pedestrian movement.

Relative to other models, social force models describes pedestrian behavior more realistically. However, they are designed to be as simple as possible. Every agent is represented by a circle with its own diameter and the model describes continuous coordinates, velocities and interactions with other objects. Each parameter has a natural interpretation, is individual for each pedestrian, and is often chosen randomly within some empirically found or otherwise plausible interval.

Social forces model describe human crowd behavior with a mixture of socio-psychological and physical forces. The most important social forces model is Helbing’s model.

**Helbing’s model:**

Pedestrians $1 \leq i \leq N$ of mass $m_i$ like to move with a certain desired speed $v_i^0$ in a certain direction $e_i^0$ and they tend to adapt their instantaneous velocity $v_i$ within a certain time
interval $\tau_i$. At the same time, the individuals try to keep a distance from other individuals $j$ and from the walls $w$ using interaction forces $f_{ij}$ and $f_{iw}$. The change of velocity in time $t$ is given by the acceleration equation:

$$m_i \frac{\text{d}v_i}{\text{d}t} = m_i \frac{v_i(t)}{\tau_i} + \sum_{j \neq i} f_{ij} + \sum_w f_{iw}$$

while the change of position $r_i(t)$ is given by the velocity $v_i(t) = \frac{\text{d}r_i}{\text{d}t}$. This model describes the psychological tendency of two pedestrians $i$ and $j$ to stay away from each other by a repulsive interaction force $A_i \exp[ (r_{ij} - d_{ij})/B_i ] n_{ij}$, where $A_i$ and $B_i$ are constants.

$d_{ij} = \|r_i - r_j\|$ denotes the distance between the pedestrians’ centers of mass, and $n_{ij} = (n^1_{ij}, n^2_{ij}) = (r_i - r_j)/d_{ij}$ is the normalized vector pointing from pedestrian $j$ to $i$. The pedestrians touch each other if their distance $d_{ij}$ is smaller than the sum $r_{ij} = (r_i + r_j)$ of their radii $r_i$ and $r_j$. If this is the case, then two additional forces are assumed inspired by granular interactions, which are essential for understanding the particular effects in panicking crowds: a ‘body force’ $k(r_{ij} - d_{ij})n_{ij}$ counteracting body compression and a ‘sliding friction force’ $\kappa(r_{ij} - d_{ij})\Delta v_{ij}t_{ij}$ impeding relative tangential motion, if pedestrian $i$ comes close to $j$.

$t_{ij} = (-n^2_{ij}, n^1_{ij})$ is the tangential direction and $\Delta v_{ij} = (v_j - v_i)t_{ij}$ is the tangential velocity difference, while $k$ and $\kappa$ represent large constants.

Finally we have:

$$f_{ij} = \left\{ A_i e^{(r_{ij} - d_{ij})/B_i} + kg(r_{ij} - d_{ij})t_{ij} + kg(r_{ij} - d_{ij})\Delta v_{ij}^t t_{ij} \right\}$$

where the function $g(x)$ is zero if the pedestrians do not touch each other ($d_{ij} > r_{ij}$), and is otherwise equal to the argument $x$.

The interaction with the walls is treated analogously. If $d_{iw}$ means the distance to wall $W$, $n_{iw}$ denotes the direction perpendicular to it, and $t_{iw}$ the direction tangential to it, the corresponding interaction force with the wall is given by:

$$f_{ij} = \left\{ A_i e^{(r_{ij} - d_{iw})/B_i} + kg(r_{ij} - d_{iw})t_{iw} + kg(r_{ij} - d_{iw})(v_i \cdot t_{iw})t_{iw} \right\}$$
Helbing’s Social Forces model [Hel00] applies repulsion and tangential forces to simulate the interaction between people and obstacles which allows for realistic ‘pushing’ behavior and variable flow rates. Helbing’s model was estimated from real data. The main disadvantage of these approaches is that agents appear to ‘shake’ or ‘vibrate’ in high-density crowds, which does not correspond to natural human behavior. There has been a considerable amount of work done using particle simulation approaches for low-density crowds. Brogan and Hodgins used particle systems and dynamics for modeling the motion of groups with significant physics [BH97]. Musse extended the social forces model to include individualism [BMO*03].

2.2.2. Cellular Automata models

Cellular automata (CA) [KNNS03, DTJ00] is an artificial intelligence approach to simulation modeling defined as mathematical idealizations of physical systems in which space and time are discrete, and physical quantities take a finite set of discrete values. A cellular automaton consists of a regular uniform lattice (2D array) with a discrete variable at each site (cells) (Figure 2). Walls and other fixed obstacles are black, while the white cells are areas that can be occupied by pedestrians.

Figure 2: 3D environment and its corresponding grid of cells.
The state of a cellular automaton is completely specified by the values of the variables at each cell. A cellular automaton evolves in discrete time steps, with the value of the variable at one cell being affected by the values of variables at the neighboring cells. The variables at each cell are updated simultaneously based on the values of the variables in their neighborhood at the previous time step and according to a set of local rules [Wol83].

Cellular automata local rules are used to describe the intelligent decision-making behavior of the automata creating and emulating actual behavior. Global emergent behavior appears from simple behavioral rules.

Each automaton is an intelligent agent in the model capable of evaluating its opportunities on a case-by-case basis. The emergent group behavior is a result of the interactions of the local rules as each pedestrian searches the available cells in its neighborhood [BA00].

Cellular automata in general provide a framework for discrete models with locally homogeneous interactions. They are characterized by the fundamental properties \((L,S,N,f)\) shown in Table 1.
A configuration $C_t : L \rightarrow S$ is a function that associates a state with each cell of the lattice. The update function $f$ changes a configuration $C_t$ into a new configuration $C_{t+1}$:

$$C_{t+1}(r) = f\left(\{C_t(i) | i \in N(r)\}\right)$$

where $N(r)$ is the set of neighbors of cell $r$, $N(r) = \{i \in L | r - i \in L\}$. This definition assumes that $f$ is deterministic, which may not the case.

Cellular-automata models [Che04, TLC*01, KNN*03], although fast and simple to implement, do not allow for contact between agents. Floor space is discrete and individuals can only move when the next cell is free. This checkerboard approach provides realistic results for lower density crowds, but unrealistic results when agents in high density situations are forced into discrete cells. Higher-level behavior can be incorporated by pre-computing paths towards high-level goals and storing them within the grid [LMM03].
2.2.3. Rule Based Models

Rule-based models [Rey87, Rey99] achieve more realistic human movement for low and medium density crowds, but cannot handle contact between individuals and therefore fail to simulate ‘pushing’ behavior. These models usually adopt a conservative approach by avoiding contact and, when densities are high, applying ‘wait’ rules to enforce ordered crowd behavior without the need to calculate collision detection and response. Cognitive models have been used in combination with rule-based models to achieve more realistic behaviors for pedestrian simulation [ST05]. Different behavioral rules can be applied to the crowd, group or individuals to achieve more believable overall crowd behavior [TMK99, OCV*02].

The most well known model to simulate life-like complex behavior is Reynolds’ local rules “boids” model [Rey87]. This model is an elaboration of a particle system with the simulated entities (boids) being the particles. The aggregate motion of the simulated flock is created by a distributed behavioral model. Each simulated agent is implemented as an independent actor that navigates according to its local perception of the dynamic environment, the laws of simulated physics that rule its motion and a set of behaviors programmed by the animator. The aggregate motion of the simulated flock is the result of the dense interaction of the relative simple behaviors of the individual simulated boids.

The basic model to simulate generic flocking behavior consists of three simple rules which describe how an individual boid maneuvers based on the positions and velocities of its nearby flockmates:

- separation: steer to avoid crowding local flockmates
- alignment: steer towards the average heading of local flockmates
- cohesion: steer towards the average position of local flockmates
Each boid has access to the whole environment description, but flocking only requires reaction within a specific neighborhood which is given by a distance (from the center of each boid) and an angle (from each boid’s direction of flight). This neighborhood can be considered as sort of limited perception. Each boid will not only avoid collision against other boids but also with obstacles in the environment.

In addition to the basic three rules used to simulate flocking, Reynolds [Rey99] (http://www.red3d.com/cwr/steer/gdc99/) introduced the more general concept of steering behaviors and placed flocking within that context. Steering behavior enhances the behaviors already presented in the original boids model by building parts for complex autonomous systems. Each of these new rules defines only a specific reaction on the simulated environment of the autonomous system.

Simple behaviors for individuals and pairs:

- Seek and flee.
- Pursue and evade.
- Wander.
- Arrival.
- Obstacle avoidance.
- Containment.
- Wall following.
- Path following.
- Flow field following steering behavior.
Combined behaviors and groups
- Crowd path following.
- Leader following.
- Unaligned collision avoidance.
- Queuing.
- Flocking.

2.3. Macroscopic models

2.3.1. Regression models

Regression models use statistically established relations between flow variables to predict pedestrian flow operations under specific circumstances. The characteristics of this flow depend on the infrastructure (stairs, corridors, etc.) [MRHA98].

2.3.2. Route choice models

Route choice models describe pedestrian wayfinding based on the concept of utility. Pedestrians choose their destinations in order to maximize the utility of their trip (comfort, travel time, etc.) [Hoo03].

2.3.3. Queuing models

Queuing models use Markov-chain models [Lov94] to describe how pedestrians move from one node of the network to another. Nodes are usually rooms, and therefore links are usually portals or doors. Markov-chain models are defined by a set of states together with transition probabilities. At each extrapolation step, a successor state is selected by either sampling from the transition distribution, or identifying the most probable successor. The state transition probabilities are estimated from the relative frequency of transitions.
between behavior prototypes observed in the training data, taking the closest behavior prototype at each time instant. Only transitions causing state change are considered.

2.3.4. **Gaskinetics**

Gaskinetics use the analogy with fluid or gas dynamics to describe how density and velocity change over time using partial differential equations [Hen71].
2.4. Models for pedestrian systems

Commercial models of pedestrian traffic are classified into microscopic and macroscopic approaches. The former studies the characteristics of individual pedestrians such as speed and interaction with other pedestrians, while the latter is concerned with groups of pedestrians rather than individual characteristics, and their analysis focuses on high density, large scale systems. Since we are interested in simulations where each agent is driven by its own goals and has its own personality and decision-making process, we will focus on microscopic systems.

In contrast, research models are more interested in creating virtual humans or animals for computer graphics and training applications that behave in an autonomous way and exhibit a wide variety of behaviors that make the virtual agents more believable. They are clearly microscopic models, but in addition address realistic body, legs, movements and graphical display realism.

In this section we will introduce a brief description of some of the most relevant systems for crowd simulation that have been developed both from industry and academia*. We conclude with a summary table showing the main features of these systems that are relevant for the work presented in this dissertation.

➢ EXODUS (Galea)

It was developed by the Fire Safety Engineering Group at the University of Greenwich. The system is able to simulate the evacuation of large numbers of individuals from large multi-floor buildings. By adopting fluid dynamic models, coupled with discrete virtual reality simulation techniques, the program tracks the trajectories of individuals as they make their way out of the building or are overcome by hazards (e.g. fire and smoke). The output of Exodus includes overall

* Additional information on other academic and commercial SW can be found at [Sti00, Tek02 and KP05]
evacuation time, individual waiting and evacuation time, and individual paths [Gal98].

- **PEDROUTE** (Buckman and Leader 1994)

  Predroute was originally developed by London Underground Limited, and has been used extensively to model crowd parameters in underground networks around the world. It is a spatial entropy maximizing model which has been used for station design, including the Olympic Railway Station, Sydney, which is designed to be capable of handling 50,000 passengers per hour. The model can simulate train and passenger movements going through a station or a building. The performance of the building is assessed using service levels, passenger densities and delays, and provides statistics of their journey times, congestion and the level of service (LOS) for each segment. Passengers are assigned to routes through the station using a dynamic assignment taking into account bottlenecks and congestion effects. Stations are divided into different blocks (e.g. like the CA models but larger and with continuous movement within them) representing stairs, escalators, platforms, ticket halls, etc., with each block having different speed of flow curves associated with the movement of pedestrians through them. The underlying assumptions and principles used in PEDROUTE are the same as other spatial interactions/entropy maximizing models and fail to incorporate the individual basic mechanisms underlying pedestrian movements. These programs cannot represent the interaction of each pedestrian with other pedestrians and the external environment, only the overall or system-wide behavior.

- **CROSSES** (Crowd Simulation System for Emergency Situations, 2002)

  CROSSES aims to provide a virtual reality tool for training people to effectively respond to urban emergency situations by using a model for generating and simulating a virtual crowd. This multi-agent system allows both scripted and
autonomous behaviors of the agents (as well as interactions among them) with the virtual environment and with the real human participants. A layered approach is used for controlling agents’ behavior which is based on a combination of rules and finite state machines [UT01].

➢ **Simulex**

Developed by P Thompson, Integrated Environmental systems, UK [TM94, TM95b]. It was developed as an evacuation model with the capability of simulating a large number of people in geometrically complex buildings. It is based on inter-person distances to specify walking speed of the individuals, and it achieves overtaking, body rotation, sideways stepping and small degrees of back-stepping. The inter-person distance is defined as the distance between the centers of the bodies of two individuals. Human body shape is represented by an elliptical body size defined by one main circle and two smaller circles bounding each shoulder.

The space is used as continuous for pedestrian movement, but discretized in order to calculate and store a distance map. The distance map is used to direct occupants to the closest available exit. The velocity of each individual depends on the distance to the people ahead.

➢ **RAMPAGE**

Particle simulation system to animate explosions and other elementary primitives; it divides human behavior into reflex reactions and decision making based on knowledge obtained from the scene. The principles for human behavior simulations is based on the Boltzmann gas equation [BC95]
EGRESS (SRT AEA TECHNOLOGY)

AEA technology started the development of Egress in 1991 [AEA02]. It is a commercial software tool for crowd simulation. The model employs artificial intelligence techniques to determine how a person would react under a variety of circumstances such as fire and smoke. The output of Egress includes evacuation time analysis, comparison between people evacuation and progression of hazard, and potential structural and procedural improvements.

The simulation is based on hexagonal grids. The approach is fundamentally a cellular automaton process in which the transition of people from cell to cell is based on occupancy of the cells. Their data is calibrated using speeds taken from experimental data [KC93].

They use the relationship:

\[ f(p) = p + v(p) \]

to obtain the flow rate, where \( p \) is the density and \( v(p) \) is the speed at crowd density \( p \).

The unimpeded mean speed of travel in a given direction is obtained from the probabilities of moving in certain directions towards the goal. The probabilities consist of:

- The probability of moving one cell closer to the goal.
- The probability of moving one cell further away from the goal.
- The probability of moving to a cell that is the same distance away from the goal.
- The probability of staying in the same place.
LEGION

Legion [LI03] was not designed as a crowd behavioral analysis system but as an investigation tool for the study of large scale interactive systems. The computational model over-simplifies the behavioral representation of individuals. First, the model employs only four parameters (goal point, speed, distance to others, and reaction time) and one decision rule (based on assumption of the least-effort principle) to represent the complex nature of individual behaviors. Furthermore, all individuals are considered to be the same in terms of size, mobility and decision-making process, and the model ignores social behaviors such as herding and leader influence [Sti00].

The Legion model works in 2D continuous space which gives more realistic paths for the pedestrians than those based on discrete grids. Legion claims that occupant movement is in agreement with extensive empirical research performed on the study of crowd movement and behavior. They have acquired and analyzed video footage of individuals and crowd behavior.

Legion states that “people’s circulation through a space is determined not only by their density but also by the specific features of the local geometry [LI03]”. Movement depends on input variables specified for each person, but also by individual’s knowledge of the environment and state of readiness, meaning interaction with signals.

STEPS

(Gwynne et al, 1999) Agent-based model with coarse grid geometry (CA). Each individual occupies one cell at any given time and moves in the desired direction if the next cell is empty. Each occupant has its own characteristics, patience factor and familiarity behavior.
In STEPS the fundamental driving mechanism for individual movement is the desire to move at a free walking speed towards the next target point in the shortest amount of time and without collision. The decision process is adhered to by every individual in the model. For each target (exit point), a potential is calculated at each grid cell on the plane. The potential value represents the distance between individual cells and the targets considering the presence of blockages (walls, columns, etc.).

At every time step (0.1 second), each target is scored based on the time of arrival. The patience level modeled into the individual or group is incorporated into the calculation, and a final score is derived. Based on the derived final score of the target, the individual located in a cell attempts eight possible directions at every time step to reach the lowest-scored target.

**ViCrowd**

ViCrowd aims at representing a model to automatically generate human crowds based on group properties instead of individuals. The individuals within a group would follow the groups’ specifications instead of their own, in order to satisfy real time requirements. It is based on Reynolds’ flocking system but includes a simple definition of behavioral rules using conditional events and reactions.

A sociological model is used to handle affinities and repulsion effects that can emerge in crowd simulation and create more complex behaviors. In this approach, control is presented through different degrees of autonomy: guided, programmed and autonomous crowd, ranging from totally interactive to totally autonomous control [MT97, MBCT98].
OpenSteer

OpenSteer provides a toolkit of steering behaviors defined in terms of an abstract mobile agent. It is the C++ implementation of Reynolds’ steering model explained in detail in section 2.2. It can be found at: http://opensteer.sourceforge.net

Massive SW

Developed by Massive Software, Inc 2005 [MS05]. This system is based on artificial life technology, using a combination of very simple rules with fuzzy logic.

It is a 3D animation tool to simulate large crowds for the special effects industry. Agents are endowed with synthetic vision, hearing and touch that allows them to react naturally to their environment. In Massive SW, reactions rely on the environment rather than an internal model and the agents respond directly to environmental stimuli, which achieves less storage than modeling the environment internally in each agent. The agents have very simple “brains”, since Massive SW has been designed to achieve realistic simulation for short periods of time (under 5 seconds) but does not deal with achieving long term goals and global navigation issues.

Reactive Navigation

Lamarche and Donikian [TD00, LD04]. Agents move within complex virtual environments represented with a hierarchical topological structure extracted from the geometry of the virtual environment to allow fast path finding as well as an efficient reactive navigation algorithm. In order to avoid collision while reaching their targets, they use an iterative optimization process. Collision prediction is based on neighborhood computations; it creates long distance neighborhood relations in sparse crowds and short distance relation in dense crowds. Collision avoidance is
achieved by predicting other agents’ position and if a collision may occur, then the agent will modify its speed and orientation vector.

➢ **Artificial fishes**

Tu and Terzopoulos [TT94] developed an artificial life approach to simulate the appearance, motion and behavior of fish in a virtual marine world and also the complex group behaviors observed in real aquatic ecosystems. Each fish behaves as an autonomous agent exhibiting fish behavior such as foraging, preying, schooling, courting and collision avoidance. These simple fish models can learn basic motor functions and perceive the environment.

➢ **ACUMEN**

Allbeck et. al. [AKA*02] developed a system for synthesizing and recognizing aggregate movements in a virtual environment with a natural language interface. Its principal components include: an interactive interface for aggregate control based on a collection of parameters extending an existing movement quality model, a featured analysis of aggregate motion verbs, recognizers to detect occurrences of features in a collection of simulated entities, and a clustering algorithm that determines subgroups.

The ACUMEN system used the PAR (Parameterized Action Representation [BSA*00]) to capture the semantics of aggregate movement for generation and recognition. Movement is based largely on a particle system-like model of group simulation, using dynamic forces acting on rigid bodies to produce the desired movement.

They extended the EMOTE features [CCZ*00] to group movement by grouping sets of aggregates verbs into classes. EMOTE was inspired by movements of observation science, in particular the Laban Movement Analysis [Mal87] and its “Effort and Shape” components.
Autonomous Pedestrians

Artificial life approach integrating motor, perceptual, behavior and cognitive components within a model of pedestrians as individuals [ST05]. The environment is represented through hierarchical data structures that efficiently support perceptual queries from the autonomous pedestrians that drive their behavioral responses and maintain their ability to plan their actions on a local and global level.

Agents perform six basic reactive behavior routines: avoid static obstacle, avoid static obstacle in a complex turn, maintain separation in a moving crowd, avoid oncoming pedestrians, avoid dangerously close pedestrians, and verify new directions relative to obstacles. Agents perform collision avoidance but not response, therefore if an intersection with another pedestrian is about to happen, which they detect by using a “front safe area”, the agent will stop, try to turn to face away and wait until space is available around its current position.
2.5. Summary

There has been a considerable amount of research on crowd or group simulation, especially in the study of evacuation dynamics. Nevertheless, there is still room for improvement.

The main focus of commercial applications is to validate their systems in terms of egress (flow rates, densities, congestion areas, evacuation times, etc.). They use either macroscopic or microscopic approaches. Since microscopic approaches are more interesting for computer graphics application, we will focus on these. The microscopic models most commonly used in industry applications are particle simulation and cellular automata models. These methods have proven to lack realism when they are applied to 3D virtual humans for animation systems, because they either look closer to particles than to real human movement (social forces model and particle simulation in general) or are restricted to checker-board configurations (Cellular Automata).

In contrast, research methods tend to focus on developing autonomous agents able to navigate large complex virtual environments while avoiding static obstacles and other agents. However, in most cases they ignore the problems that arise when dealing with very high density crowds. These systems usually apply some sets of rules to avoid collision based on modifying the speed or trajectory of the agents. Consequently these models are sufficient for medium and low density crowds, however when the crowd is very dense, they yield unnatural emergent behavior such as individuals stopping and waiting for space to clear up. There is no concept of body-to-body contact leading to pushing agents in a crowd, or individuals being dragged by the crowd. Further effects such as falling, injury, incapacitation and others walking over the fallen agent are also ignored. Our system has as its main goal to deal with all these features that emerge in real high density crowds.

In terms of global navigation, the published systems for crowd simulation assume that agents have complete information about the environment. An agent can access the entire internal structure of the environment and use algorithms such as A* (several techniques
have been exploited to achieve real time when performing global path planning for large groups of agents), or else the environment is discretized as a grid that stores potentials or distance maps that the agents will follow locally to reach the goal.

It is essential to provide an agent with the ability to explore partly known environments and learn new features. Another crucial aspect of crowds that is ignored elsewhere is that people have the ability to communicate with others in order to exchange information.

Our approach focuses also on improving the realism of the agents’ behavior by allowing them to have partial information about the environment and be able to extend their memory (or mental maps) as they explore the environment and communicate with other individuals in the crowd. Agents can also exhibit different behaviors based on different roles.

Finally, some psychological factors need to be incorporated with the purpose of modifying the overall performance of an individual based on its mental state.

Table 2 shows a comparison of some of the most significant models in crowd animation. The table emphasizes the main features in multi-agent simulations and pedestrians evacuations in order to compare the contributions of our model to others.
Table 2: Comparison of different systems for animation of large groups.

<table>
<thead>
<tr>
<th>System</th>
<th>Collision response</th>
<th>Particles shaking corrected</th>
<th>Behavior Method</th>
<th>Communicate or signals</th>
<th>Individual or roles</th>
<th>Real-time</th>
<th>Learning</th>
<th>Spatial structure for motion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social forces (Helbing)</td>
<td>Yes</td>
<td>no</td>
<td>Forces</td>
<td>no</td>
<td>some</td>
<td>yes</td>
<td>no</td>
<td>cont.</td>
</tr>
<tr>
<td>Rule based</td>
<td>No</td>
<td>not needed</td>
<td>Rules</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>cont.</td>
</tr>
<tr>
<td>CA</td>
<td>No</td>
<td>not needed</td>
<td>CA</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>2d grid</td>
</tr>
<tr>
<td>Simulex</td>
<td>Yes</td>
<td>no</td>
<td>Distance maps</td>
<td>no</td>
<td>some</td>
<td>no</td>
<td>no</td>
<td>cont.</td>
</tr>
<tr>
<td>Egress</td>
<td>No</td>
<td>not needed</td>
<td>CA</td>
<td>no</td>
<td>some</td>
<td>no</td>
<td>no</td>
<td>hexagonal grid</td>
</tr>
<tr>
<td>ViCrowd</td>
<td>No</td>
<td>not needed</td>
<td>Rules+</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>cont.</td>
</tr>
<tr>
<td>OpenSteer</td>
<td>No</td>
<td>not needed</td>
<td>Rules</td>
<td>no</td>
<td>some</td>
<td>yes</td>
<td>no</td>
<td>cont.</td>
</tr>
<tr>
<td>Legion</td>
<td>No</td>
<td>not needed</td>
<td>Least-effort</td>
<td>some</td>
<td>some</td>
<td>no</td>
<td>no</td>
<td>cont.</td>
</tr>
<tr>
<td>Exodus</td>
<td>No</td>
<td>not needed</td>
<td>CA</td>
<td>no</td>
<td>some</td>
<td>no</td>
<td>no</td>
<td>2d grid</td>
</tr>
<tr>
<td>Steps</td>
<td>No</td>
<td>not needed</td>
<td>CA</td>
<td>no</td>
<td>some</td>
<td>no</td>
<td>no</td>
<td>2d grid</td>
</tr>
<tr>
<td>Massive SW</td>
<td>No</td>
<td>not needed</td>
<td>Rules+</td>
<td>fuzzy logic</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>cont.</td>
</tr>
<tr>
<td>Reactive Navigation</td>
<td>No</td>
<td>not needed</td>
<td>Rules</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>cont.</td>
</tr>
<tr>
<td>Artificial fishes</td>
<td>No</td>
<td>not needed</td>
<td>Rules</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>cont.</td>
</tr>
<tr>
<td>ACUMEN</td>
<td>Yes</td>
<td>no</td>
<td>Particle sim.</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>cont.</td>
</tr>
<tr>
<td>Crosses</td>
<td>No</td>
<td>not needed</td>
<td>Rules+</td>
<td>FSM</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no cont.</td>
</tr>
<tr>
<td>Autonom. Pedestrians</td>
<td>No</td>
<td>not needed</td>
<td>Alife</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>cont.</td>
</tr>
<tr>
<td>MACES+HiDAC</td>
<td>Yes</td>
<td>yes</td>
<td>Extended social forces</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>cont.</td>
</tr>
</tbody>
</table>
2.5.1. Some limitations of the current commercial software for crowd evacuation

Since most of the commercial tools for crowd evacuation are based on the cellular automata model (e.g: STEPS, EXODUS, EGRESS, etc.), it is important to understand their movement simulation artifacts. We also describe in this section some of the limitations that current commercial software tools have in terms of simulating human psychology and physiology. Andersen [ABG05] provides a more detailed discussion regarding the limitations of grid-based pedestrian simulation models.

**Grid Size**: Using a cellular automata model and therefore having a discrete grid for the simulation creates several limitations. Some of the main problems that occur are fixed densities and unrealistic flow rates through portals. Grid size becomes a crucial parameter to calibrate in order to achieve the desired behavior. Individuals will move with their desired velocity unless all of the cells around them are occupied or blocked, which causes the person to wait for the next empty cell in the desired direction of movement.

Having a fixed grid size limits the maximum densities achievable. For example, if the grid size is defined as 0.5m² the maximum density at any time will be 4 persons/ m², while the literature in crowd behavior reports densities of 7.4 persons/ m² where people can still move [ABG05].

A second issue arises when the grid is not accurately aligned with the geometry. This can lead to the appearance of artifacts where only one person at a time can get through a door in the simulation, when in reality the door size is big enough to fit two people crossing simultaneously. An example of this situation can be observed in Figure 4.
**Fatigue factor** is not included in the simulation. The speed values given in the literature are based on those collected during fire drills and normal situations. During an actual fire evacuation in a high rise building, slower speeds when walking downstairs have been reported. This can be the result of fatigue when walking downstairs for long periods of time (unfit, older or disabled people). Fatigue yields people needing rest stops which can then provoke bottlenecks.

**Speeds in stairwells:** In some reported scenarios (e.g. WTC attack) the observed speeds during the real evacuation descending stairwells was 0.2m/s which is half the slowest speed given by egress when walking downstairs. The reason why these speeds occurred was because there was an ascending counterflow of firefighters that was blocking the downstairs flow. In 1 meter wide stairs, it should be possible to have two flows of people moving in opposite directions, but since one of those flows contains fire fighters carrying all their gear, it turns out that one of the flows needs to completely stop in order to let the other move.
**Route selection:** Path finding in grid based models consists of traversing the centers of squared cells. Distances between centers can be stored before the simulation takes place. The method is usually based on “potential maps” which identifies a discrete approximation of the shortest path towards the destination and stores this information in the cells in order to achieve an efficient simulation. The main problem that potential maps have is that they favor 45 degrees diagonal movement, and the resulting routes are not always realistic. Figure 5 shows the unnatural paths followed by the people in Exodus (grey paths) compared to some of the real paths that should have been followed if the space was continuous (red dotted lines).

![Figure 5 Paths followed based on potential maps in Exodus](image)

Potential maps computed on grids have the following problems [ABG05]: they yield highly unrealistic space utilization, cannot guarantee equivalence on return trips which artificially segregates opposite flows, and distort path length and thus pedestrian travel time.
Uneven use of stairwells occurs due to familiarity or initial distance to exits, which leads to different utilization of the stairs. Because distances are computed before the simulation and route selection is based on the potential maps, some stairwells may attract more individuals than others. This can have a large impact on the overall evacuation time. This can be a positive emergent behavior if it matches with what would actually occur in the real building, but how to validate the positive impact of this behavior in the accuracy of the results is not clear.

![Figure 6 Uneven utilization of stairwells during route selection (STEPS)](image)

*Figure 6 Uneven utilization of stairwells during route selection (STEPS)*
2.6. Navigation

Coordinating the movement of groups of agents plays an important role to simulate swarms of robots, animals and pedestrians in computer graphics and civil engineering applications. Most research focuses on techniques for modeling individual behavior of flocks inspired by Reynolds’ boids [Rey87] and Helbing’s social forces models [HFV00] when moving in continuous space and Cellular Automata when dealing with discretized grids.

For continuous space, Rule Based models and Social Forces models can be sufficient for simple environments where agents cannot get locked in local minima, however they will have difficulties when simulating larger and more complex environments. In order to navigate a complex environment, we need to have some high-level representation of the environment. Among the most popular techniques for crowd navigation are cell and portal graphs [LCC06, PB06, PLT05] potential fields [Che04, Gal98, TM94], and roadmaps [KSLO96, BLA02, SKG05].

In Cellular Automata models navigation can be performed through grid based search using A* algorithms, potentials, or flow tiles. Computer games have commonly used A* search to group motion [RN94, LK05]. In this approach the environment is divided into a heterogeneous grid and the search is based on expanding towards the most promising neighbor of already visited positions. Although A* can find the shortest path to a goal and several improvements have been added to achieve fast solutions [Rab02], it is still necessary to run the algorithm again to find a new path for each new goal and for each agent in the group. Alternatively, potentials or flow tiles preprocess the required path information and then store it within each cell, so during the simulation, each agent will query the cell for navigation information.
2.6.1. Cell and Portal Graphs

Cell and portal graphs are often used to abstract away the geometry and calculate navigation by solving the problem of getting from one node of the graph to another through a sequence of nodes and portals [PB06, PLT05]. When employed for indoor scenes, nodes usually represent the rooms defined by their enclosing walls and portals correspond to the doors. On top of that partition, an adjacency graph is built where each portal connects the two rooms on both sides of the door. Outdoor environments can also be represented with cell and portal graphs where cells are pedestrian pathways and portals appear between pedestrian pathways and crossings [LCC06].

2.6.2. Flow tiles and potential field methods

In potential field methods, the environment is discretized into a regular grid. Then a potential is associated with each cell which corresponds to the sum of a repulsive potential generated by obstacles in the environment and an attractive potential generated by the goal. Therefore, gradient methods can be applied to find a path from any origin in the environment to a goal position. The method has problems, such as local minima where the individuals could get stuck and never reach the goal [Gal98, TM94, LMM03].

Flow tiles offer a similar approach [Che04]. Tiling can be constructed to meet a wide variety of external and internal configurations. Each flow tile contains a small vector field, and the combination of several tiles can produce large flows.

2.6.3. Probabilistic Roadmaps (PRMs)

PRMs have been widely used in robotics and navigation for autonomous agents. The basic idea of Probabilistic Roadmaps (PRMs) [KSLO96] consists of computing a very simplified representation of the free space by sampling configurations at random. Then the sampled configurations are tested for collision and each collision-free configuration is
retained as a “milestone”. Each milestone is linked by straight paths to its k-nearest neighbors. Finally the collision-free links will form the PRM.

Several groups have used PRMs to perform navigation for large groups of autonomous agents. Bayazit et. al. [BLA02] simulate crowds with various group behaviors like homing, shepherding and exploring combined with PRMs to drive the characters towards a goal or to explore a scene; their main motivation is to expand flocking behaviors by endowing the agents with some global information about the environment. Lien et. al. [LRM*05] extended that work by introducing multiple shepherds and allowing them to coordinate without communication and then incorporated dynamic roadmaps by modifying edge weights as an implicit means of communication between flock members [BLA05]

2.7. Environment Modeling

Haumont et. al. presented an algorithm for volumetric cell-and-portal generation for indoor scenes based on an adaptation of the 3D watershed algorithm [HDS03]. The watershed is created using a flooding analogy in the distance field space. Flooding starts from local minima, and each minimum produces a region (room). Portals appear where regions meet during the growth. The algorithm automatically classifies each room as a cell and the openings (doors and windows) as portals, generating the cell and portal graph of any indoor environment.

Pettré et. al. [PLT05] introduced navigation graphs for multilayered and uneven terrain based on some motion planning methods from robotics [HST99, Che99]. In this approach the space is divided into free-space and obstacles to be avoided. A Voronoï diagram of the free-space is calculated and then collision-free convex cells are built along the diagram. The navigation graph is obtained from the adjacency graph of the cells. The novelty of this work is in extending the basic navigation graph to multilayered terrain by classifying some free-space areas as obstacles based on the slope of the terrain.

Lerner et. al. presented a method to efficiently create cell and portal graphs for both interior and exterior environments [LCC06]. The algorithm input is a set of half edges in
2D that can be extracted from the geometry. They use a two pass algorithm: the first step creates an initial partition and then the second step refines it. Their algorithm heuristic strives to create small portals as a means for generating an effective partition. The method supports incremental changes of the model by locally re-computing and updating the partition.

Shao et. al. represent virtual environments by a hierarchical collection of maps [ST05]: (i) a topological map that represents the connections between different parts of the virtual world; (ii) perception maps, that provide information regarding perceptual queries; and (iii) path maps which enable online path-planning for navigation. The topological map contains nodes corresponding to the environmental regions and edges representing accessibility between regions. The path maps include a quadtree map which supports global, long-range path planning and a grid map which supports short-range path planning.
Chapter 3

Building Modeling

The navigation of crowds of autonomous agents in complex environments requires having an efficient abstract representation of the virtual environment where the agents can rapidly perform wayfinding. This abstract representation can also be used to store some precomputed information about the environment that will speed up the navigation and also be helpful to achieve fast perception for local motion computation. Our system (MACES+HiDAC) can handle two types of environment generation. The first generates a maze-like building environment from input parameters (dimensions, number of exits and number of hazards) (Figure 7). The second creates a building (Figure 8) from a floor plan editor (Appendix A).

Figure 7: Example of a maze used for our experiments with 2 exits and 8 hazards
For any given environment our system automatically generates a cell and portal graph where each cell represents a room and portals correspond to doors. The stairwells are treated as cells with two portals, one at each end of the stairwell.

Each room contains information about the list of walls in that room and the list of static obstacles, so when the agents need to perform local motion within the room, they will query the room for those lists of static obstacles against which they need to perform collision detection.

In order to achieve real-time interactive navigation, some other relevant information about the environment is pre-calculated and stored. Among the information stored are: paths towards the exits, distances from each door to a destination point, and the position of the attractors that will be used during the local motion to steer the agents. In order to save space, this environment information is also considered internal knowledge of the agents, however since our agents can have different roles, and therefore exhibit diverse behaviors, they will have access to different levels of information which will provide diversity in their level of knowledge about the environment.
3.1. **Cell and Portal graph automatic generation**

Our system receives as an input an arbitrary building model and creates a cell and portal graph, identifies all the walls and obstacles that belong to each cell, and stores that information within the cell. The building geometry is represented by a grid decomposition that contains different elements representing walls, doors, obstacles, stairs, windows, etc.

From that representation, an intermediate 2D grid is created for each floor, where the value 0 will be assigned to grid cells representing free-space, –1 will be assigned to those grid cells containing a wall, -2 assigned to grid cells occupied by doors, and other negative values are assigned in the same manner for holes and stairs. This 2D grid is not employed in the final movement of the agents as the Cellular Automata models do, but instead it is used as an intermediate step to obtain the geometrical information of the building.

The algorithm proceeds in four steps:

- generate the cell and portal graph for each floor.
- identify stairs and link floors through new cells.
- identify and store walls.
- identify and store obstacles.

The cells in the graph generated correspond to the rooms and determine the continuous space in which the virtual autonomous agents will perform navigation.

3.1.1. **Generate cell and portal graph for each floor**

Once we have the grid decomposition where 0 indicates free space and negative numbers indicate non-empty space (doors, stairs, obstacles and doors), we start an iterative conquering process starting from the top left corner cell that is empty. We assign a positive
number to this cell that will represent the room ID in the cell and portal graph, and then this ID is propagated using a breadth-first traversal. The propagation of the cell ID continues until the entire room is bounded by cells having either 0 (wall) or -1 (door). The following procedures show the algorithm in detail:

**Table 3: Algorithm to assign cell IDs to free space**

```plaintext
Procedure find_cells (f)
    cellID := 1
    for f={0…maxFloors}
        for i={0..maxX}
            for j={0…maxZ}
                if grid[f][i][j] = free_space
                    create_cell(cellID)
                    cellID := cellID + 1
                    flood_neighborhood(cellID,f,x,z)
            end_if
        end_for
    end_for
end_for
```

**Table 4: Algorithm to propagate IDs**

```plaintext
Procedure flood_neighborhood (cellID,f,x,z)
    if (grid[f][i][j] = free_space)
        if (((x≥0) and (x<maxX) and (z≥0) and (z<maxZ))
            flood_neighborhood(cellID,f,x-1,z)
            flood_neighborhood(cellID,f,x+1,z)
            flood_neighborhood(cellID,f,x,z-1)
            flood_neighborhood(cellID,f,x,z+1)
        end_if
    end_if
end_if
```

Where:
cellID: is the positive number representing the room ID

f: is the floor number

x and z: are the coordinates of the cell in the 2D grid representation for that floor.

Figure 9 shows (a) the initial geometry of the building for one floor and (b) the corresponding grid decomposition where walls, doors and obstacles have been identified. Figure 10 shows how the algorithm to identify cells propagates the IDs.

Figure 9: Building geometry (a) and grid partition with walls, obstacles and door (b)

Figure 10: Eight first steps of the cell identification and ID propagation algorithm.
Once all the cells have been identified, we need to generate the cell and portal graph by joining the rooms through the doors. This is carried out by traversing the grid representation from left to right, top to bottom, looking for doors. When a door is found, a portal is created that will join the two cells appearing at both sides of the door. Two attractor points will be associated with each portal. These attractors are located at both sides of the door (Figure 13) and will be used for local motion of the autonomous agents as steering points.

Once all the doors have been detected and the portals created, we obtain the final cell and portal graph for the floor (Figure 11)

![Cell and Portal graph](image)

**Figure 11: Cell and Portal graph**

### 3.1.2. Identify stairs and link floors through new cells

If we have a multi story building, then the algorithm will create the cell and portal graph for each floor, and then the stairs will be included as new cells that have portals with the lower and upper floors.

After creating all the subgraphs that correspond to each floor of the building, the algorithm traverses the grid searching for stairwells. When a stairwell is found, a new cell and two new portals are created. One portal will link the stair cell with the lower floor and the other portal will link it to the upper floor.
Figure 12 shows the cell and portal graph of a two story building with one stairwell at the left bottom corner of the building. Cell 41 in the graph corresponds to the stair.

3.1.3. Identify and store walls

Once the cell and portal graph has been generated, we need to create the list of walls corresponding to each room. From the grid cell decomposition generated after creating the cell and portal graph, we will have information about walls (cells with ID=0) and room IDs (since each free-space cell now contains the ID of the corresponding room in the cell-and-portal graph).

The grid is traversed sequentially from left to right, top to bottom, searching for walls along the X axis, and then from top-bottom, left-right searching for walls long the Z axis. Walls are delimited by corner walls and doors. For each sequence of cells marked with 0’s, between those delimiters, a wall in the environment will be created and assigned to the rooms at both sides of the wall. The result of the wall-finding algorithm can be observed in Figure 13. For clarity of the results, each of the walls has been colored to match
the color of the cell. In the image we can also see the attractor points located centered in each doorway and displaced slightly (0.7m) into each room.

![Figure 13: Walls assigned to each room and door attractors](image)

### 3.1.4. Identify and store obstacles

Finally the algorithm searches for obstacles in the environment and assigns them to the room based on the ID of the adjacent neighboring cells in the 2D grid decomposition. The obstacles can be observed in Figure 13.
3.2. Pre-calculating data for real-time simulation

The autonomous agents in the crowd need to interact with the environment in order to avoid obstacles and implement global planning. To achieve fast perception of the static obstacles in the environment, every cell has a list of walls and objects. For each wall we store the equation of the plane that defines the wall, with the normal of that plane pointing towards the interior of the room, and also the ending points of that wall segment. For every obstacle, we store its position and dimensions (obstacles are bounded by cylinder or oriented bounding boxes, OBBs). The last elements to be stored are the two door attractors. These attractors are used to steer the agent in a natural manner when crossing portals.

When an agent needs to move from one location to another in a virtual room, it will query the room for the lists of walls and obstacles, and then calculate collision detection with those objects, while moving from the previous attractor to the next attractor.

In order to perform global navigation, we also need to store information about paths within the building from each cell to each of the exits in the building. Each cell will contain one or more alternative paths to each exit. During the simulation, agents will be able to query the room for different types of information, depending on their roles.

At each room the information stored will contain for each exit the sequence of rooms and doors and the overall distance in meters that correspond to the shortest path. Shortest path information is calculated by performing a breadth-first traversal starting at the exit cells and propagating to adjacent rooms. Figure 14 shows an example of the breadth first-traversal algorithm, indicating the propagation order through the rooms with decreasing intensity of color.
The breadth-first traversal starts from the room that has an exit and pushes it in a queue. The algorithm proceeds iteratively, popping an unvisited node from the queue, visits it, marks it as visited, adds its neighbors to the queue and repeats it until all the nodes in the graph have been visited. As the traversal advances, it updates the distances from each portal to the exit portal, and if a cell is reached through several doors, then the algorithm compares the previous shortest distance stored against the new one, and only if the new distance is shorter, the node will be pushed in the list. When two cells are connected through several doors, the one with the minimum global distance to the exit is chosen as part of the shortest path.

The details of the algorithm are shown in Table 5
**Procedure** propagate_shortest_paths

for each exit node E
  push_list(E)

  while list non empty V=pop_list
    mark V as visited
    for each neighbor room N
      if not discovered N
        if N already had a path stored then
          if newGlobalDist < prevGlobalDist
            modify stored shortest path
            push_list(E)
          end_if
        else
          push_list(E)
        end_if
      end_if
    end_for
  end_while
end_for

*Table 5: Algorithm for shortest paths propagation.*

If only shortest paths were stored, then when an agent would find the desired path towards an exit blocked, it would need to perform some search in the environment in order to find a different route. Agents with limited knowledge about the environment may not be able to find in their mental maps an alternative route and therefore need to explore the environment. But for those agents with complete knowledge about the environment, a search in the complete cell and portal graph representing the building should be feasible. In order to speed up the simulation time, we calculate this information offline and store it in the environment. This represents the knowledge of the agents and can be shared among
those with trained roles. The alternative paths are calculated by modifying slightly the previous algorithm. In the search for alternative paths, when a node is reached through a different route, first of all the algorithm checks that the current node is not already contained in the path towards the exit to avoid cycles. Then if it is not already in the path, this new path is stored in the node as an alternative route, and propagated to all the neighboring nodes except for the one from which it arrived.

The algorithm finishes when all the possible alternative routes have been propagated and stored. Then the list of alternative paths is ordered by distances, with the aim of speeding up the query process from the agents during simulation time.

The following table shows the alternative paths stored for the current building:

<table>
<thead>
<tr>
<th>Node</th>
<th>Origin</th>
<th>Shortest path</th>
<th>Alternative paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(2,4)</td>
<td>(2,3,7,5,4)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(4)</td>
<td>(3,7,5,4)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(2,4)</td>
<td>(7,5,4)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>{}</td>
<td>{}</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>(4)</td>
<td>(7,3,2,4)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>(7,5,4)</td>
<td>(7,3,2,4)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>(5,4)</td>
<td>(3,2,4)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>(7,5,4)</td>
<td>(7,3,2,4)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>(4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>(4)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node</th>
<th>Origin</th>
<th>Shortest path</th>
<th>Alternative paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>(8,7,5,4)</td>
<td>(8,7,3,2,4)</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>(4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>(15,4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>(17,4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>(4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>(17,4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>(4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>(17,4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>(4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>(18,17,4)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 6: Alternative paths example*
Chapter 4

Framework

We have developed a framework for high density multi-agent simulation with a bottom-up approach. On the low level agents move within a room driven by a social forces model with psychological and geometrical rules affecting several parameters that will allow for a wide variety of emergent and high density behaviors. Above the motion level we need a wayfinding algorithm that will perform navigation in large complex virtual buildings, using communication and roles to allow for different types of behavior and navigation abilities. Both the motion level and the wayfinding with communication and roles can be affected by psychological factors that are initially given as personality parameters for each agent, but also can be modified during the simulation to affect an agent’s behavior.

Agents move within complex virtual environments with several rooms, corridors, obstacles, stairwells and doors that can be opened or closed at any time during the simulation (Figure 15). In order to navigate these virtual environments, route selection is carried out through an interactive high level way-finding algorithm that dynamically calculates the global path based on the agents’ knowledge of the environment [PB06].
In order to achieve real-time interactive navigation, some relevant information about the environment is pre-calculated and stored as explained in chapter 3. Among the information stored are paths towards the exits, distances from each door to a destination point, and the position of the attractors that will be used during the local motion to steer the agents. Agents will have access to this information based on their roles. This allows us to represent different levels of knowledge about the environment, but any other information required by the agent will have to be gathered through exploration, learning and communication with other agents.

The high-level navigation process is interactive, meaning that agents are endowed with a decision making process that will allow them to follow the known route or make new decisions based on changes in the environment and their psychological parameters.

Changes in the environment include a door appearing locked which makes that path invalid or creates a bottleneck in some part of the desired path. These
changes make it more difficult to reach a goal and therefore, based on the level of impatience assigned to the agent, a decision could be made to take a different route.

Each cell of the building stores the shortest path to each exit. There are two ways in which this information can be interpreted. On the one hand we can consider that this shortest path stored in the cell corresponds to the path that an agent in that cell would have followed when entering the building and therefore is the only one known. On the other hand, we could consider this shortest path as being the one indicated by the exit signs in a building and therefore would be the ones that everyone would follow in case of emergency.

Agent spatial knowledge is represented by a graph where the nodes are the rooms and the arcs are the portals between rooms. This mental graph that represents the memory of the agent will have more nodes added as it navigates and explores the building. At any time, each agent needs to know which rooms of the building have been fully explored and which others still have portals that lead to rooms that have not been visited yet. The mental graph abstracts away the actual geometry of the environment. The building geometry is used later to compute locomotion transit times and portal bottlenecks.

Another crucial source of information is communication with other agents. There are two pieces of information shared by the agents every time two or more agents meet in a room: locations of hazards found in the building that are blocking some of the paths, and parts of the building that have been fully explored by other agents and found to have no exit through them. This localized sharing of mental models is the key to our algorithm’s wayfinding behavior.

Each individual within the crowd will have different behaviors depending on two attributes: leadership and training:

- Leaders and trained agents have complete knowledge about the internal building structure that would also help others during the evacuation process. An example of this type of agent would be a firefighter.
• Leaders but untrained agents correspond to people that by nature can handle stress better, tend to help others and will explore the building searching for new paths.

• Non-leaders and untrained (followers) represent dependent people who might panic during an emergency situation, and reach the point where they are incapable of making their own decisions.

4.1. Interaction between high level, low level and psychological models

MACES is a multi-agent system without a centralized controller. Each agent has its own behavior based on personality variables that represent physiological and psychological factors observed in real people. Agent behaviors are computed at two levels:

• High-level behavior: navigation, learning, communication between agents, and decision-making.

• Low-level motion: perception and a set of reactive behaviors for collision avoidance, detection and response in order to move within a room.

Figure 16 shows the interaction between the two levels. The High-Level module receives information about bottlenecks and door changes that have been perceived by the agent and makes decisions based on that information and its current knowledge of the environment. Once the high-level decides the next room to walk to, it sends the next attractor point to the Low-Level module to carry out the required motion to reach it. When the Low-Level module reaches the attractor, it queries the High-Level module for the next attractor in its path towards the destination.

The Motion sub-module queries the Perception sub-module about positions and angles of obstacles, crowd density ahead of the agent, and velocity of dynamic obstacles. Based on information perceived and the internal state of the agent (current
behavior, panic, impatience, etc.), the Motion sub-module calculates the velocity and next position of the agent, and sends a message to the Locomotion sub-module to execute the correct feet movements.

Both high level and low level behavior are affected by a module representing the psychological and physiological attributes of each agent. The idea of using a psychological model is that agents will operate independently in perceiving the simulated world and in forming their reactions to it. At no point will they be prescribed or programmed via rules or procedures. We only model personality attributes and individual agents will make their own decisions that lead to the emergent crowd behavior.

The high-level is affected by changes in psychological elements such as panic or impatience, by altering the decision-making process (e.g. an impatient agent will select a different route after perceiving congestion in a door). Other elements such as an agent’s memory and orientation abilities can be affected by high level behavior (psychological studies show that a person under panic may suffer disorientation). Finally an agent’s psychological state may trigger changes in roles (e.g. a leader changing to follower when its panic level gets very high or a trained agent exhibiting untrained behavior when suffering from disorientation).
The low-level is also affected by changes in the psychological state of the agent which will trigger modification of the agent’s speed, probability to fall, pushing thresholds, etc. The psychological model needs to have as input information about environment events detected by the agent’s perception system and information obtained through communication. Then this information will be combined with the agent’s current emotional state in order to modify it if necessary and send back the right input to both low level and high level modules.

In Figure 17 we can observe how the psychological model interacts with the navigation and local motion elements. This psychological module contains information regarding the agent’s state and psychological factors that are currently affecting its behavior. This module needs to supervise both high level and low level in order to detect changes that should alter the internal state of the agent and then apply the corresponding modification at both decision-making and local motion levels.

![Figure 17: Architecture overview with psychological model](image-url)
4.2. Parameters affecting crowd behavior

The following table shows the parameters that can be input in MACES to specify initial conditions for a simulation and psychological and physiological personality attributes for the agents. In the current framework, those parameters are specified by the user through an interface and can be modified during the simulation. It could also be possible to get those values through an API from a high level psychological model that would drive the internal emotional state of the agents [POS*05].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leadership</td>
<td>Percentage</td>
<td>Percentage of leaders in the crowd (the rest will be dependent individuals)</td>
</tr>
<tr>
<td>Trained</td>
<td>Percentage</td>
<td>Percentage of trained (building knowledgeable) individuals among the leaders</td>
</tr>
<tr>
<td>Communication</td>
<td>Boolean</td>
<td>Whether agents can communicate</td>
</tr>
<tr>
<td>Panic</td>
<td>Percentage</td>
<td>Percentage of people that will exhibit panic when an alarm goes off or a hazard is perceived</td>
</tr>
<tr>
<td>Panic propagation</td>
<td>Percentage</td>
<td>Percentage of people with high probability of exhibiting panic behavior when perceiving other agents in panic</td>
</tr>
<tr>
<td>Impatience</td>
<td>Percentage</td>
<td>Percentage of people that will avoid bottlenecks when other paths are available</td>
</tr>
<tr>
<td>Falling</td>
<td>Percentage</td>
<td>Percentage of people with high probability to lose equilibrium under severe pushing (representing physical abilities)</td>
</tr>
<tr>
<td>Pushing Threshold</td>
<td>Percentage</td>
<td>Percentages for each distance allowed from other agents for which repulsion forces will not apply</td>
</tr>
<tr>
<td>Right preference</td>
<td>Percentage</td>
<td>Percentage of people that will tend to move towards the right when facing opposite flow</td>
</tr>
<tr>
<td>Avoidance</td>
<td>Percentage</td>
<td>Percentage for each magnitude indicating how abruptly a person will try to avoid others by walking around instead of forming lines during normal conditions.</td>
</tr>
</tbody>
</table>

*Table 7: Parameters affecting behavior.*
The current interface allows the user to create either the entire population at once and have each parameter being distributed among the entire crowd according to the percentage assigned, or if the user desires to have more control over the individual parameter of the agents, then smaller groups of agents can be created with specific personality attributes. For example, if the user wants a population of 40 agents, where 50% are leaders with maximum pushing threshold and the other 50% are dependent agents with minimum pushing threshold, then the user should first create a segment of 20 agents, with 100% leadership and 100% max pushing threshold. Next, add another segment of 20 agents with 0% leaders and 100% min pushing threshold.

Figure 18 shows the interface used to create the segments/population. As we can see, the interface allows the user to specify the size of the segment and the percentages of each parameter that will affect that group of agents. The user can create as many segments as desired.

Figure 18: MACES+HiDAC interface
**Leadership**

Specifies the percentage of agents in the crowd that tend to be leaders and take decisions in terms of global navigation when they find themselves blocked due to a hazard or a locked door. The rest of the individuals are considered dependent people, which means that in the situation of not knowing where to go, they would rather follow others than explore the environment by themselves.

**Trained**

Among the leaders population, there will be a percentage that will have complete knowledge about the internal connectivity of the building, and therefore if the shortest path being followed becomes invalid they will immediately know an alternative solution. Basically their internal mental map corresponds to the cell and portal graph representing the environment, while the rest of the agents will only have a subgraph of it at any given time, which will expand as they explore and communicate with other agents.

**Communication**

This can be set to true or false based on whether we want the agents to have the ability to communicate or not during the simulation. Communication is the process that allows agents to exchange relevant information about the environment, such as “there’s a fire in that room” or “the door on the left leads nowhere”, etc.

**Panic**

Initial percentage of people that will tend to panic when an alarm goes off or when they see a hazard.
Panic propagation

Percentage of people that even though they will not start to exhibit panic behavior when the alarm goes off, may change to panic mode after seeing others panic for some amount of time, or by having many individuals around them pushing for a certain period of time. This is a very interesting feature that allows our system to exhibit emergent panic propagation that will affect bottlenecks and flow rates through portals.

Impatience

Overall percentage of people that when observing a bottleneck in their next portal, may reconsider their selection and interactively change their path if they know an alternative short route.

Falling

In order to represent the fact that some individuals are more likely to fall when they find themselves in a high density crowd (elderly, disabled or weaker people, etc.) we allow the user to represent this factor by setting a percentage of people that are likely to have equilibrium problems. For example, in the case where the user needs to simulate an evacuation from a building with 80% of elderly individuals, this variable can be used to represent the likelihood of some of those people having difficulties in maintaining their equilibrium when being pushed by others.

Pushing thresholds

Pushing thresholds identify the distance that agents are willing to maintain to other agents of the crowd, i.e. the “contact distance” between individuals. It can be set to very small (0), average (1), or large (2). When an agent falls inside that distance, it will provoke a repulsion force that pushes it away from the other agent. Pushing behaviors can vary in a crowd, where some individuals are more likely to try to open their way through a high density crowd even if it is at the cost of pushing others.
**Right preference**

When people walk in a crowd they tend to apply social rules that usually match driving rules. In many countries people drive on the right and therefore when they are walking and another person is moving in the opposite direction, social rules will make each human try to avoid the other by slightly diverting their paths towards their right hand side. This parameter is used to set the percentage of people that will exhibit right preference.

**Avoidance**

Avoidance factor is linked to collision avoidance behavior. Collision avoidance deals with applying forces that alter the agent’s trajectory in order to smoothly avoid static and dynamic obstacles. The avoidance factor gives the strength of those forces which requires agents to do more or less abrupt direction changes. The result affects mainly the width of any line/queue that arises during normal conditions. We establish three values (weak, medium, strong) and the percentage of each of them among the population.
Chapter 5

Low level: Local Motion

Local agent motion is based on the social forces model introduced by Helbing [Hel00], but we also apply geometrical information and psychological rules to enable a wider variety of behavior closer to that of real people. At the same time we use additional forces to eliminate some of the unnatural behavior that Helbing’s model exhibits when simulating 3D humans instead of 2D dots.

We will briefly introduce Helbing’s model and then give a detailed list of psychological (panic, impatience) and geometrical (distance, areas of influence, relative angles) rules that we have incorporated into our model to eliminate unrealistic artifacts and to allow new emergent behaviors:

- Prevent agents from appearing to vibrate
- Natural bi-directional flow rates
- Organized behavior – queuing
- Pushing through a crowd
- Agents falling and becoming obstacles
- Panic propagation

These emergent behavior are driven by the parameters given in Table 7
5.1. Introduction

In terms of defining the motion of each agent, we classify three main approaches: social forces systems, rule based models and cellular automata models. None of these models, however, can realistically animate high density crowds. HiDAC focuses on the problem of simulating high-density crowds of autonomous agents moving in a natural manner in dynamically changing virtual environments. In this section we will explain how psychological and geometrical rules are layered on top of the basic social forces model in order to improve high-density crowd movement and add realism. Since applying the same rules to all agents leads to homogeneous behavior, agents are given different psychological (e.g., impatience, panic) and physiological (e.g., speed) traits that trigger heterogeneous behaviors based on crowd density and personality.

Each agent is endowed with perception and reaction to static and dynamic objects and agents within the current room. Common perception approaches in the literature are based on casting a set of rays to calculate intersections. We introduce a simpler approach to perceive the environment and make decisions, while still achieving highly realistic results.

Realistic movement is achieved both in terms of collision avoidance and collision response. Over longer distances tangential forces gently steer the agent around obstacles, while over shorter distances repulsion forces are applied to enable collision response. Pushing behavior is achieved by varying the long/short pushing threshold of each individual. Agents in a hurry (moving fast) and with small pushing thresholds will not respect others’ personal space and will appear to push their way through the crowd. In contrast, agent with large pushing thresholds (more ‘polite’) will respect lines and wait for others to move first.

Each agent scans an ellipse-shaped region in front of them. Relaxed agents temporarily stop when another agent moves into their path, while impatient agents do not respond to this feedback and tend to ‘push’. Our model stops impatient agents from appearing to ‘vibrate’ as they try to force their way through dense crowds, as we
add temporal braking forces to the social force model. These forces only apply when repulsion forces fall within a specified range of angles opposing forward motion. The angles are set based on agent personality and crowd density.

5.2. Agents’ speeds and densities

This section describes the quantitative factors that can be utilized to estimate the pedestrian movement accurately.

There is a large amount of data in the civil engineering and fire evacuation literature to calculate the movement component of total evacuation time. In order to simulate real pedestrians’ movement, there are several moving parameters to consider such as:

**Speed**: rate of travel along a corridor, ramp and stairwells.

**Flow**: number of persons passing a particular segment of the egress system per unit of time.

**Specific Flow**: flow per unit width of the egress component (persons/second per meter of doorway width).

Most of this information on the movement of people, including disabled individuals, has been collected through fire drills, in stairs, corridors and through doorways. In order to accurately simulate human behavior, we employed the data available in “The SFPE Engineering Guide to Human Behavior in Fire” 2002 [SFPE02].

Speed is a function of the density of the occupant flow, type of egress component and mobility capabilities of the individual. For a density greater than 0.55 pers/m

\[ v = k - a kD \]  

Eq. 1

and for densities less than 0.55 pers/m, there are not enough people around an individual to impede its walking speed, therefore maximum walking velocities are defined by:

\[ v = 0.85 \cdot k \]  

Eq. 2
where:
\[ v = \text{speed in meters/sec} \]
\[ a = \text{constant (0.266 m}^2/\text{pers}) \]
\[ k = \text{velocity factor, as described in Table 8 [SFPE02]} \]
\[ D = \text{density of occupant flow, pers/ m}^2 \]

<table>
<thead>
<tr>
<th>Egress component</th>
<th>K (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corridor, doorway</td>
<td>1.40</td>
</tr>
<tr>
<td>Stair, riser = 190 mm</td>
<td>1.00</td>
</tr>
<tr>
<td>Stair, riser = 272 mm</td>
<td>1.08</td>
</tr>
<tr>
<td>Stair, riser = 165 mm</td>
<td>1.19</td>
</tr>
</tbody>
</table>

*Table 8: Velocity factors*

At lower densities, individuals can move freely in the environment being able to reach their desired maximum speed and at higher densities velocity will be reduced. Mean velocities for impaired individuals and people without disabilities given by Shields, et al., [STS96] are presented in Table 9

<table>
<thead>
<tr>
<th>Impairment</th>
<th>Level Walkway (m/s)</th>
<th>Stairwells-down (m/s)</th>
<th>Stairwells-up (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electric wheelchair</td>
<td>0.89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manual wheelchair</td>
<td>0.69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crutches</td>
<td>0.94</td>
<td>0.22</td>
<td>0.22</td>
</tr>
<tr>
<td>Walking Stick</td>
<td>0.81</td>
<td>0.32</td>
<td>0.34</td>
</tr>
<tr>
<td>No disability</td>
<td>1.24</td>
<td>0.70</td>
<td>0.70</td>
</tr>
</tbody>
</table>

*Table 9: Mean velocities*

It is also important to stress the importance of density when simulating high density crowds. Real measurements show that crowds can maintain fluid movement even at densities of 7pers/m² [BBQ*05] or in some extreme situations, real densities observed have reached up to 13.5pers/m² as reported by Tsuji [Tsu03]. Reduction in
walking speed is already noticeable for densities above 2pers/m² and congestion appears for densities of 4pers/m².

Fruin [Fru71] introduced concept of level of service where flow rate is expressed as a function of density. According to Fruin pedestrian area occupied ranges from about 0.5 to 2.3 m²/pers on walkways and from 0.4 to 0.9 m²/pers on stairways. The corresponding flows range from 0.38 to 1.37, as indicated in Table 10.

<table>
<thead>
<tr>
<th>Fruin Level of Service</th>
<th>Density Pers/m²</th>
<th>Space m²/pers</th>
<th>Flow rate (pers/m/s)</th>
<th>Av. speed (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>&lt; 0.31</td>
<td>&gt; 3.22</td>
<td>&lt; 0.38</td>
<td>1.3</td>
</tr>
<tr>
<td>B</td>
<td>0.43 - 0.31</td>
<td>2.32 - 3.24</td>
<td>0.38 - 0.55</td>
<td>1.25</td>
</tr>
<tr>
<td>C</td>
<td>0.72 – 0.43</td>
<td>1.39 – 2.32</td>
<td>0.55 – 0.82</td>
<td>1.15</td>
</tr>
<tr>
<td>D</td>
<td>1.08 - 0.72</td>
<td>0.93 – 1.39</td>
<td>0.82 – 1.10</td>
<td>1.00</td>
</tr>
<tr>
<td>E</td>
<td>2.17 – 1.08</td>
<td>0.46 – 0.93</td>
<td>1.10 – 1.37</td>
<td>0.7</td>
</tr>
<tr>
<td>F</td>
<td>&gt; 2.17</td>
<td>&lt; 0.46</td>
<td>&gt; 1.37</td>
<td></td>
</tr>
</tbody>
</table>

*Table 10. Fruin levels of Service*

In our system, agents are given an initial maximum speed following a normal distribution with mean 1.24 (standard deviation=0.2) for rooms and mean 0.7 (standard deviation=0.2) for stairwells. The maximum speed can be increased though when an agent is in panic, where a running speed will apply (normal distribution with mean = 1.7 m/s and standard deviation = 0.2) (Table 11). As the density increases, individuals will reduce their speed due to interaction and repulsion forces with other agents and obstacles in the environment. Individuals can also fall as a consequence of the pushing behavior in high density crowds, and latter stand up again to continue with their movement when the area around them clears. Fruin levels of service have been used as a reference to calibrate our system in order to achieve realistic flow rates through doors (up to 1.9 pers/m/s. Table 12) and realistic densities, although higher densities than the ones described by Fruin have been allowed for panic situations as indicated in the literature [BBQ*05, TRB94].
### Table 11: summary of walking speeds according to several studies [TM95a]

<table>
<thead>
<tr>
<th>Study</th>
<th>Old people Walking speed (m/s)</th>
<th>Young people Walking speed (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Slow</td>
<td>Normal</td>
</tr>
<tr>
<td>Men</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blanke &amp; Hageman</td>
<td>1.38</td>
<td></td>
</tr>
<tr>
<td>Himann et al</td>
<td>1.21</td>
<td>1.47</td>
</tr>
<tr>
<td>Women</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finley et al</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>Blanke &amp; Hageman</td>
<td>0.32</td>
<td></td>
</tr>
<tr>
<td>Ferrandez et al</td>
<td>0.82</td>
<td>1.08</td>
</tr>
<tr>
<td>Himann et al</td>
<td>0.89</td>
<td>1.14</td>
</tr>
<tr>
<td>Leiper &amp; Craik</td>
<td>0.96</td>
<td>1.15</td>
</tr>
<tr>
<td>O'Brien et al</td>
<td>0.74</td>
<td>0.97</td>
</tr>
<tr>
<td>Both</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cunningham et al</td>
<td>1.05</td>
<td>1.33</td>
</tr>
<tr>
<td>Elble et al</td>
<td>0.94</td>
<td>1.39</td>
</tr>
<tr>
<td>Waters et al</td>
<td>0.81</td>
<td>1.22</td>
</tr>
<tr>
<td>Judge et al</td>
<td>1.06</td>
<td>1.43</td>
</tr>
</tbody>
</table>

### Table 12: Maximum and Ultimate flow rates [TM95a]

<table>
<thead>
<tr>
<th>Source</th>
<th>Maximum Design flow (pers/m/s)</th>
<th>Ultimate Flow Capacity (pers/m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fruin</td>
<td>1.37</td>
<td></td>
</tr>
<tr>
<td>Predtechenskii &amp; Milinskii</td>
<td></td>
<td>1.83</td>
</tr>
<tr>
<td>Daly</td>
<td>1.43</td>
<td></td>
</tr>
<tr>
<td>SPFE handbook</td>
<td>1.3</td>
<td></td>
</tr>
<tr>
<td>Handkin &amp; Wright</td>
<td>1.48</td>
<td>1.92</td>
</tr>
<tr>
<td>Polus et al</td>
<td>1.25</td>
<td>1.58</td>
</tr>
<tr>
<td>Ando et al</td>
<td>1.8</td>
<td></td>
</tr>
</tbody>
</table>
5.2.1. Walking speeds and densities when walking downstairs

Figure 19 shows the relationship between speed and density and Figure 20 shows the relationship between flow and densities both for downstairs movement. This information has been used to calibrate the agents’ movement when walking downstairs.

**Figure 19:** Relationship between speed and density walking downstairs during evacuation [SFPE02]

**Figure 20:** Relationship between flow and density walking downstairs [SFPE02]
5.3. Perception

Autonomous agents need to perceive the environment to avoid static and dynamic obstacles while walking between two attractors. HiDAC provides efficient perception by using the cell and portal graph. As the agents walk around the environment, the lists of dynamic objects within each room are rapidly updated. Therefore an agent can obtain the necessary data by queries to the cell.

For each obstacle we need to calculate its distance and, if it is close enough to the agent, the angle between the agents’ desired direction and the line joining the center of the agent and the obstacle. The distance and the angle provide enough information to establish how relevant that obstacle is to the trajectory. While the agent looks for possible obstacles, it also updates its perceived density of the crowd ahead which will be important in the decision-making process.

Humans can perceive a binocular field of view (FOV) from 120° to 180°, the latter being the most common. We can simulate human perception by having the virtual agents be only aware of those objects falling within a specified angle from their direction of movement (assuming the head is oriented in the same direction). Currently our system is set to detect objects falling in a FOV of 180° (90° right and left of the direction of movement). This is calculated from the dot product between the direction of movement vector and the vector joining the current position with each object in the room. Since the dot product gives us the cosine of the angle between the two vectors, if that value is bigger than 0 it means the object falls within the agent’s FOV.

Objects within our FOV are perceived but only objects falling within a rectangle area ahead of us are relevant in terms of obstacle avoidance. Figure 21 shows an agent (A) perceiving several obstacles simultaneously. In reality, we do not give obstacles avoidance preferences based on distance, but on how much they affect our desired trajectory. In Figure 21 we can observe that even though the wall and the column are closer to the agent, our algorithm also factors in angles, which makes the agent (B) ahead the most important obstacle at this moment.
Since our algorithm only needs distances and angles, it is faster than casting rays for intersection with every obstacle, since our method has cost $O(N)$, where $N$ is the number of obstacles in the room, while ray casting has cost $O(R \cdot N)$, where $R$ is the number of rays cast and $N$ the number of obstacles. The visual results achieved for our crowd simulation prove our method to be sufficient for an agent’s environment perception used to make decisions regarding its motion.

5.4. Basic Social Forces model

Helbing’s social forces model describes human crowd behavior with a mixture of socio-psychological and physical forces. Pedestrians $1 \leq i \leq N$ of mass $m_i$ like to move with a certain desired speed $v_{i0}$ in a certain direction $e_{i0}$ and they tend to adapt their instantaneous velocity $v_i$ within a certain time interval $\tau_i$. At the same time, the individuals try to keep a distance from other individuals $j$ and from the walls $w$ using interaction forces $f_{ij}$ and $f_{iw}$. The change of velocity in time $t$ is given by the acceleration equation:
This model generates realistic phenomena such as arching in the portals [PM71] and the “faster is slower” effect. [ES93] (Figure 22)

In our model the desired velocity direction within each room is given by an attractor point that is located close to the next portal the agent needs to cross. This behavior orients the agent so that its velocity is radially aligned towards the target (attraction point). In the absence of obstacles or other agents, every agent will flow along the evacuation direction field (passing through the portals unobstructed). Floor is treated as a continuum, not as a discrete regular grid.
Collision detection is performed only with the people within the same room, except when people are crossing a portal. In this situation care must be taken to avoid intersection between agents leaving and agents entering the room. Our approach to this problem consists of keeping track of the people currently crossing a portal. When an agent is near a door, collision detection is performed not only with the other agents in the room but also with those currently crossing the doorway (geometrically located close to the attractors at both sides of the door).

Figure 24 shows the different states in which an agent can appear while crossing a door and the transition between states. In order to walk from cell N to cell N+1, an agent will have A as its first attractor point. When the agent’s position is within half a meter from A, then the high level algorithm will set B as the next attractor. In this state the agent will be inserted in the list of current agents crossing the portal. When the agent gets close to attractor B, the high level algorithm will decide the next door based on the agent’s knowledge about the environment and the desired goal. In the figure the next attractor will be A’. The agent will stay in the list of agents currently
crossing until he moves half a meter away from B, and therefore there will be no risk of intersection with agents crossing the portal.

Figure 24: Crossing Portals States

In Figure 25 we can appreciate how the agents’ state changes as they walk through the door. The colored dots above each agent represent the state as indicated in Figure 24.

Figure 25: Agents crossing a portal
5.5. HiDAC system

5.5.1. Extending the social forces model

Our extended social forces models need to consider collision detection and response when agents are in contact with other agents or obstacles, and also collision avoidance when an obstacle appears further away than the contact distance but within the desired trajectory.

The total force applied to an agent is given by:

\[
 f_{\text{Total}} = f_{\text{Attract}} \cdot W_{\text{Attract}} + f_{\text{Walls}} \cdot W_{\text{Walls}} + f_{\text{Obst}} \cdot W_{\text{Obst}} + f_{\text{desiredSpeed}} + f_{\text{OtherAgent}} \cdot W_{\text{OtherAgent}} + f_{\text{FallenAgents}} \cdot W_{\text{FallenAgents}} 
\]  
**Eq. 4**

The forces affecting agents’ movement are:
- \( f_{\text{Attract}} \): Force towards the attractor (determines the agent’s desired direction of movement)
- \( f_{\text{Walls}} \): Force to avoid walls.
- \( f_{\text{Obst}} \): Force to avoid non-traversable obstacles other than walls (e.g. columns)
- \( f_{\text{FallenAgents}} \): Force to avoid obstacles that an agent will try to avoid but can walk over (e.g. person on the floor)
- \( f_{\text{OtherAgents}} \): Force to avoid other agents (can be avoided or pushed away from trajectory, depending on current behavior)

\( f_{\text{Walls}}, f_{\text{Obst}} \) and \( f_{\text{OtherAgents}} \) can be either a tangential force to steer around an obstacle or a response force to avoid intersection, depending on the distance to the other agent/obstacle.

\( f_{\text{FallenAgents}} \) will always be a tangential force. The reason is that an agent on the floor is a type of obstacle that we will try to avoid but if the density of the crowd gets very high and we are being pushed, it is a possibility that we will step over that
person, since it is not a non-traversable obstacle such as a wall. More information about how this obstacle is treated can be found in section 5.5.6.

$W_{\text{Attrac}}, W_{\text{Walls}}, W_{\text{Obst}}, W_{\text{OtherAgents}}$, and $W_{\text{FallenAgents}}$ are weights that indicate how much each of these elements should affect the total force $f_{\text{Total}}$. $W_{\text{OtherAgents}}$ can be specified by the user through the interface, and the possible values are: minimum=0.1, medium=0.5, maximum=1.0. If any of those elements in the environment were so close to the agent that repulsion forces become necessary, then the weight for that particular force would be 0, which indicates that it has to be treated as a response. Finally, $f_{\text{desiredSpeed}}$ is the force that tries to maintain the current desired direction to smooth the trajectory when reacting to other obstacles in the environment.

The new desired position for the agent at time $t+1$ is given by:

$$p_d(t+1) = p(t) + v(t) \cdot \frac{f_{\text{Total}}}{\|f_{\text{Total}}\|} \quad \text{Eq. 5}$$

where the direction of movement is given by the unit vector in the direction of $f_{\text{Total}}$ and the magnitude is such that the movement of the pedestrian has the desired walking velocity $v(t)$. When an agent’s new desired position overlaps with any static or dynamic object in the same room, collision response applies to prevent that intersection from occurring. The final position, $p_f$, for the agent at time $t+1$ is calculated by applying the repulsion force to the desired position $p_d(t+1)$

$$p_f(t+1) = p_d(t+1) + \bar{r} \quad \text{Eq. 6}$$

where $\bar{r}$ is the repulsion force that depends on the obstacle being intersected and the distance from it, as we will discuss next.
5.5.1.1. Calculating Repulsion Forces

Obstacles in the environment are:

<table>
<thead>
<tr>
<th>Object</th>
<th>Walls and columns</th>
<th>Other walking Agents</th>
<th>Fallen Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>Non-traversable</td>
<td>Can be pushed</td>
<td>Can walk over (or fall over if being pushed)</td>
</tr>
<tr>
<td>Priority</td>
<td>Highest</td>
<td>Medium</td>
<td>Lowest</td>
</tr>
</tbody>
</table>

*Table 13: Objects classification and repulsive priorities*

Each of these types of obstacles needs to be treated differently in terms of which repulsion forces apply and how. We also need to give them different priorities in order to distinguish which obstacle has preference when several obstacles may be intersected simultaneously. A *Non-traversable* obstacle should never be overlapped. The second type *Can be pushed* has lower priority than *Non-traversable* meaning that it will create a repulsion force, however if it was necessary an agent could overcome the strength of that force and instead push the other agent or squeeze it slightly. Since we are dealing with non-deformable 3D human figures, we cannot physically squeeze a virtual human, but we can allow a very small overlap between agent spaces that would simulate squeezing (0.015m of allowed overlapping).

Finally the last type of obstacles named *Can walk over* represent agents that have fallen. The rest of the agents will try to walk around a fallen individual, but if the density was very high and an agent is being pushed, then there is the possibility of walking over the agent. This type of obstacle will not produce repulsion forces, it will create instead tangential forces. We will explain this type of obstacle avoidance in detail in section 5.5.6.
**Repulsion Forces with other Agents**

Repulsion forces between agents only apply when one agent intersects the personal space of the other agent (understanding personal space as pushing threshold parameter, which represents the minimum space between agents before the two body will collide). Since not all the agents have exactly the same value for their personal distance, it is possible that the repulsion forces will not be reciprocal and only one of the agents will be pushed away.

The personal space of each agent is defined by a radius that will be slightly larger than the agent’s waist radius. Figure 26 shows the radius defining the personal distance, which is given by the waist radius $r_i$ and the pushing value $\xi_i$. (The pushing threshold is given in Table 7. This value can be minimum=-0.05, medium=0 or maximum=0.05. Notice that for minimum pushing threshold against an agent with minimum or medium value, the agents will slightly overlap. This overlap can be seen as “squeezing” if we were using deformable bodies for the agents.) The repulsion force depends on the distance between the actor and other pedestrians surrounding it.

*Figure 26: Personal space of an agent. Showing a case where one is compromised but not the other.*
The repulsion force from agent $j$ to agent $i$, is given by the following equation:

$$\vec{f}_{ji}(t) = \frac{(p_i(t) - p_j(t)) \cdot ((r_i + \xi_i) + r_j - d_{ij}(t))}{d_{ij}(t)}$$  \hspace{1cm} \text{Eq. 7}$$

where

$$d_{ij}(t) = \|p_i(t) - p_j(t)\|$$  \hspace{1cm} \text{Eq. 8}$$

The magnitude of the repulsion force depends on the intrusion or overlap of the personal space radius of the agent being pushed ($r_i + \xi_i$) and the waist radius of the agent pushing ($r_j$). As we can observe in Figure 26, since the pushing thresholds ($\xi$) for different agents can vary, it is possible to have non reciprocal repulsion forces between agents.

The magnitude of the repulsion force is directly proportional to the overlap between the two agents’ personal space.

The final repulsion force affecting an agent is given by the sum of all the repulsion forces from agents being currently overlapped:

$$\vec{f}_{Agents}(t) = \sum_j \vec{f}_{ji}(t)$$  \hspace{1cm} \text{Eq. 9}$$

**Repulsion Forces with Walls**

Collision response from walls applies when the personal space of an agent ($r_i + \xi_i$) overlaps with a wall. Overlapping occurs when the distance from the center of the agent to the wall is smaller than the personal distance. This distance is calculating using the plane equation that defines the plane:
Figure 27: Repulsion force with the walls.

\[
\bar{f}_{wi}(t) = \frac{\bar{n}_w \cdot ((r_i + \xi) - d_{iw}(t))}{d_{iw}(t)}
\]

Eq. 10

where \( d_{iw}(t) \) is calculated by substituting the center of the agent position \( p_i(t) \) in the plane equation of the wall:

\[
d_{ij}(t) = p_x \cdot n_x + p_y \cdot n_y + p_z \cdot n_z + D
\]

Eq. 11

The magnitude of the repulsion force is also directly proportional to the overlap between the agent’s personal space and the wall (it is also inversely proportional to the distance between the agents).

5.5.1.2. Calculating Avoidance Forces

When no repulsion forces apply, an agent moves towards its goal optimizing the movement by taking the best path to the target location while avoiding other agents and obstacles. Avoidance forces are achieved through tangential forces that will drive an agent away from other nearby agents or obstacles that are not close enough to overlap with its personal space. Tangential forces are necessary when a possible
collider appears ahead of the agent and within a certain distance of its desired path. In this section we explain how the tangential forces are calculated and applied for one obstacle. If there were several obstacles ahead of the agent, then priorities apply to decide which one needs to be avoided first as we will explain in more detail in section 5.5.3.

Figure 28 shows the area that affects an agent’s perception and triggers avoidance behavior. When the distance $d_j$ from an obstacle or agent $A_j$ to agent $A_i$ velocity vector $\vec{v}_i$ is smaller than the sum of the two radii $r_i$ and $r_j$, then avoidance forces are needed to modify the trajectory of $A_i$ around $A_j$.

![Figure 28: Avoidance forces with agents and round obstacles](image)

The tangential force that will steer $A_i$ is obtained from:

$$f_{\text{avoid}} = \left\| \left( \overrightarrow{w_{ji}} \times \vec{v}_i \right) \times \overrightarrow{w_{ji}} \right\|$$

Eq. 12

Figure 29 draws the corresponding situation when a wall appears on the desired path of the agent and within a distance that represents the influence area for avoidance.
The avoidance force to steer the agent away from the wall is the tangential force:

$$f_{avoid} = \left\| \vec{n}_w \times \vec{v}_i \right\{13}{ alternatives}$$

The magnitude of the avoidance force is inversely proportional to the distance between the agent and the obstacle or wall; the smaller the distance, the more abrupt the change in the trajectory direction.

The magnitude also depends on the personality of the agent (avoidance parameter from Table 7), which allows us to calibrate the system in order to achieve different emergent behaviors. For example, when simulating normal conditions, where individuals need to line up and wait to get through a door, the values given to the agents affecting the avoidance magnitude (min=0.1m, medium=0.5m, max=2m) will lead to wider or thinner line formation.

This magnitude value is represented in Eq.1 by $W_{Walls}$, $W_{Obst}$ and $W_{OtherAgents}$ depending on the obstacle being avoided.
5.5.1.3. Putting all the forces together

In the absence of other agents or obstacles, the movement of an agent is driven from its current position \( p(t) \) to the attractor point \( a(t) \), is an attractor force that makes the agent path a straight line with unit direction given by \( f_{\text{Attrac}} \):

\[
f_{\text{Attrac}} = \frac{a(t) - p(t)}{\|a(t) - p(t)\|}
\]  

Eq. 14

Since in this situation all the other forces in Eq.1 will be equal to 0 (notice that \( f_{\text{desiredSpeed}} \) will be equal to \( f_{\text{attrac}} \) since in the absence of obstacles, the agent moves in a straight line). The final movement for the pedestrian is only dependent on the attractor force and the agent will move with constant speed equal to its maximum speed.

When the agent needs to avoid obstacles, but not to apply collision response, then Eq.1 applies. The final total force will be the one steering the agent. In order to perform collision avoidance of distant obstacles, it is only necessary to modify the direction of the movement, but not the magnitude of the velocity.

When collision response is necessary, the algorithm proceeds as indicated in Table 14.
Procedure calculate_next_step

if no obstacles, walls or agents within the perception area
\[ W_{Walls} = 0, \quad W_{Obst} = 0, \quad W_{OtherAgents} = 0, \quad W_{FallenAgents} = 0 \]

else
  if walls or obstacles need to be avoided
    \[ W_{OtherAgents} = 0, \quad W_{FallenAgents} = 0 \]
  else
    if agents need to be avoided
      \[ W_{Walls} = 0, \quad W_{Obst} = 0, \quad W_{FallenAgents} = 0 \]
  endif
endif
endif
endif
endif

\[ f_{Total} = f_{Attract} W_{Attract} + f_{Walls} W_{Walls} + f_{Obst} W_{Obst} + f_{desiredSpeed} + f_{OtherAgents} W_{OtherAgents} + f_{FallenAgents} W_{FallenAgents} \]

\[ p(t+1) = p(t) + v(t) \cdot \frac{f_{Total}}{\|f_{Total}\|} \]

if collision response needs to be applied against walls
\[ p(t+1) = p(t+1) + f_{rep_{walls}} \]
else
  if collision response needs to be applied against obstacles
    \[ p(t+1) = p(t+1) + f_{rep_{obst}} \]
  else
    if collision response needs to be applied against fallen agents
      \[ p(t+1) = p(t) + v(t) \left( \frac{f_{Total}}{\|f_{Total}\|} \cdot 0.2 + f_{FallenAgents} \cdot 0.8 \right) + f_{rep_{otherAgents}} \]
    else
      if collision response needs to be applied against agents
        \[ p(t+1) = p(t+1) + f_{rep_{otherAgents}} \]
      endif
    endif
  endif
else
  if collision response needs to be applied against walls
    \[ p(t+1) = p(t+1) + f_{rep_{walls}} \]
  endif
endif
endif
endif

Table 14: Procedure to calculate the next position
5.5.2. Solution for the “shaking” problem in high-density crowds.

When an agent encounters a bottleneck in a high-density crowd, applying the basic social forces model leads to an unnatural continuous vibration behavior. This undesirable behavior is eliminated in HiDAC by having braking forces. The magnitude of these braking forces depends on the density of the crowd, personality of the agent (whether the panic behavior parameter is above 5, on a scale from 0=non panic to 10=extreme panic), and current situation (global alarm on or off).

Braking forces are applied when repulsion forces appear within a specified angle of the agent’s desired direction of movement. This follows the psychological idea that when a person is in a very dense crowd, and repulsion forces are pushing backwards, unless the person is in an extremely panicked situation, she will stop pushing temporarily until some space is available to move forward. Only forces directed backward are relevant (Figure 30), since slowing down or simply waiting can reduce them. If the forces appear to be towards our desired direction, we cannot decrease their intensity by not moving forward and therefore no reaction is necessary.

\[
\begin{align*}
v &= \text{desired direction} \\
r &= \text{repulsion force} \\
\Theta &= \text{angle}(v, r) \\
\text{If } \cos(\Theta) > 0 \quad \text{then braking forces apply}
\end{align*}
\]

*Figure 30: Example of repulsion forces which are necessary to apply braking forces.*

This method succeeds in reducing shaking behavior, while still allowing body contact and therefore pushing behavior. Since braking forces do not apply when we
are being pushed forwards, this achieves the desired result of people appearing to be pushed through the doorways when there is a high-density crowd behind them.

Table 15 shows the algorithm to apply repulsion forces.

```plaintext
if there are repulsion forces from other agents
  dotProd = \( f_{OtherAgents} \cdot v(t) \)
  if (dotProd < -0.9)
    stopShaking = rand()%30
  else if (dotProd < -0.7)
    stopShaking = rand()%20
  else if (dotProd < -0.4)
    stopShaking = rand()%10
  else
    if (stopShaking > 7)
      stopShaking = 7
    else
      stopShaking = stopShaking – 1
  end if
end if
if (alarm)
  stopShaking = stopShaking - panicLevel
end if
if \(|f_{OtherAgents}| > \kappa_{forceAgents}\) = 0
else
  \( \delta = 0.5 \)
end if
\( p(t+1) = p(t) + v(t) \cdot \frac{f_{total}}{f_{total}} \delta + f_{rep\_otherAgent} \)
else
  stopShaking = -1
  \( p(t+1) = p(t) + v(t) \cdot \frac{f_{total}}{f_{total}} \)
end if
```

Table 15. Braking forces algorithm
The dot product between the sum of repulsion forces from other agents and the desired velocity vector, indicates the cosine of the angle between those two vectors. If the cosine of the angle is smaller than 0, it means that the vectors are facing in opposite directions. The closer that the dot product is to -1 (collinear vectors facing opposite directions) the longer the time for the braking forces. The variable stopShaking reflects the length in time during which braking forces will apply. If the alarm is on, then the total time will be reduced depending on the panic level of the agent.

If the magnitude of the repulsion force is larger than the threshold $\kappa_{\text{forceAgents}}$ then the braking factor $\delta$ will be equal to 0, otherwise it will be 0.5. The braking factor determines the intensity of the braking force applied. If $\delta=0$ then the braking force has the same magnitude as the final movement force, which will have the effect of stopping the desired movement, although the agent can still change position if being pushed. If $\delta = 0.5$ then it will halve the desired movement distance.

### 5.5.3. Overtaking and bi-directional flow behavior

In order to exhibit realistic counter flows and overtaking behavior, we expand the social forces model to include tangential forces that depend on density of the crowd, individual behaviors and distance to obstacles. This approach allows us to simulate human behavior by setting parameters related to real human movement.

Avoidance behavior is triggered when there are other obstacles or agents within a certain distance of the desired velocity vector, but not close enough to require collision response.

The parameters that affect the tangential forces for obstacle avoidance are:

- Distance to obstacles
- Direction of other agents relative to our desired velocity vector.
- Density of the crowd
Agents perceive a rectangular area ahead of them (Figure 31). If an intersection occurs with that rectangle, then tangential forces will be applied with the right orientation in order to slightly modify the direction of movement and make a curve in the trajectory to avoid collision.

The angle between two agents’ velocity vectors determines whether their movements are confluent or opposed. This angle is also used to simulate human decision-making of how to react to an imminent collision. For example, if someone is walking on the left side of a corridor, and another person walks towards him/her on the right, neither of us will change direction, but if both are walking in the middle of the corridor, then people have a tendency to move towards their right side. Therefore, when the velocity vectors are almost collinear, the tangential forces will steer the agent to its right hand side for those agents that exhibit right preference. This parameter is user-specified as right preference in Table 7. This parameter specifies the percentage of people that will exhibit right preference during the simulation.

In Figure 31 agent $A_0$ detects agent $A_1$ and agent $A_2$ as possible obstacles. Each agent produces a repulsion force ($r_{10}$ and $r_{20}$) toward $A_0$. $A_1$ is farther away than $A_2$, but since it is moving against $A_0$, the perception algorithm establishes this obstacle as having higher priority.

![Figure 31: Collision Avoidance rectangle of influence](image-url)
The tangential force $\tilde{f}_t$ is calculated that will steer $A_0$ to avoid $A_1$. Next, the normalized tangential vector is multiplied by two scalar weights:

$$\tilde{f}_t' = \frac{\tilde{f}_t'}{\|\tilde{f}_t\|} \cdot W_d \cdot W_o \quad \text{Eq. 15}$$

where $W_d$ is the weight due to the distance between agents, and $W_o$ is the weight due to the difference in orientation of the velocity vectors.

$W_d$ increases exponentially as the distance between the two agents becomes smaller. The agent $A_0$ trajectory will change more abruptly as the distance to $A_1$ decreases. $W_o$ only distinguishes whether the perceived agent is moving in the same direction as $A_0$ (and therefore unless $A_0$ is walking faster, there will be no need to change the $A_0$ trajectory) or against $A_0$ (and so we need a higher weight for the tangential forces that will steer $A_0$ appropriately).

The last parameter to consider is the crowd density, which each agent perceives at any given time. If the crowd is very dispersed, then people look for obstacle avoidance from far away and keep their preference for the right hand side of the space; but when the crowd is very dense, then the right preference is not so obvious and several bi-directional flows can emerge. Modifying the length of the collision avoidance rectangle and reducing the angle for right preference based on perceived density achieve this behavior. Figure 32 shows different bi-directional flow-rate formation for low and high densities.
People with blonde hair are initially on the right and walk towards the left, while dark-haired people move from the left to the right. (a) low-density flows, (b) high-density without altering the viewing rectangle and right preference, (c) high-density with HiDAC.

Figure 32b shows the result if the length of viewing rectangle and right preference parameters are not affected by density. The emergent behavior shows an unrealistic “triangle” of people moving in opposite directions and, later in the simulation, two perfectly formed groups of people appear to move in opposite directions, which is less common in real high-density crowds.

An interesting emergent behavior that appears in our simulation of high-density crowds (Figure 32c), is the formation of lanes of people moving in the same direction among a crowd moving in the opposite direction. This is a behavior that is often observed in real crowds, and it emerges in our model even though it is not explicitly implemented.
Animation Results on Bi-directional flows and overtaking animation

Natural bi-directional flows and overtaking behavior emerges through the combination of variable areas of influence, right preference, and relative direction between autonomous agents. In this section we show some additional snapshot of several animations where these type of behavior emerges.

Figure 33, shows an example with low densities where agents smoothly modify their trajectories to avoid collision with other agents.

![Figure 33: Results on bi-directional flows for low densities](image)

Figure 34 shows results of an animation with high densities where movement is more limited, and thus agents tend to change their desired paths more abruptly as soon as enough space is available to maneuver around the other agents. Avoidance preferences are set based on relative direction, which causes agents to avoid opposite traffic instead of overtaking people moving in their same direction when space is too limited to avoid both types of traffic.

In Figure 34 we can observe two different animations. The left sequence of images shows the self-organizing behavior that emerges when the influence areas are reduced as the density of the crowd increases, thus providing more reactive behavior. It also exhibits the outcome of reducing the right preference as the density increases.

On the other hand, the sequence of images on the right shows the emergent behavior when those elements are not applied. The outcome is that the crowd slowly
splits into two perfect lanes of individuals moving in opposite directions on their right hand side which look unrealistic.

The sequence on the left exhibits a more realistic high density crowd behavior, where several interleaved lanes of individuals walk in opposite directions. The autonomous agents in this simulation will try to avoid immediate collision with agents walking in the opposite direction, which leads to the natural outcome of line formation almost as if they were following the person in front of them. This behavior, although not explicitly programmed, emerges as the result of the agent ahead of them clearing space by making the opposite traffic spread out in order to avoid collision. Once the agents have walked passed the dense opposing traffic, they will start exhibiting once again more overtaking behavior. They will also re-spread along the width of the corridor driven by tangential forces to avoid others and thus maximize the clear space ahead.
Figure 34: High density crowds bi-directional flows based on right preference
Overtaking behavior emerges when a faster agent is walking behind a slower agent and there is no immediate oncoming traffic. Figure 35 gives an example of this behavior. The agent A (marked with a red circle) is moving faster than the agent right in front of him. For the first five screen shots of the animation, since there is oncoming traffic within A’s rectangle of influence, those oncoming agents have avoidance preference over the ones walking in the same direction. After all the oncoming traffic has walked by, the slow agent right in front of A will get preference when setting the avoidance forces and consequently A will initiate the overtaking maneuver.

*Figure 35: Example of overtaking animation.*
5.5.4. Organized behavior - queuing

When a “normal” situation is being simulated, people will respect lines and wait for others to walk first. This type of organized behavior is achieved by adding influence disks ahead of each agent that drive the temporal stopping behavior.

Figure 36 shows the area for each agent that triggers a waiting behavior when another agent falls within it. In reality people feel more comfortable at closer distances when they are facing in the same direction as opposed to standing face-to-face, therefore the radius of these areas depends on the angle between the desired direction and the closest agent velocity:

\[ \alpha = \text{angle}(v_0, v_1) \]

If \( \cos \alpha < 0 \) then the agent is walking in the opposite direction, so a larger radius for the waiting area is necessary. If \( \cos \alpha \geq 0 \) then both agents are moving in the same direction, therefore a smaller radius is needed.

The radius also depends on the type of behavior desired (panic vs normal) and the panic value assigned to the agent; e.g., agitated or panicking agents will not respect these distances, therefore \( r=0 \), for agents will levels of panic from 0 (no panic) to 4 the thresholds are:

<table>
<thead>
<tr>
<th>Panic value</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waiting radius (in meters)</td>
<td>Moving in same dir</td>
<td>0.5</td>
<td>0.4</td>
<td>0.3</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>Moving in opposite dir</td>
<td>0.25</td>
<td>0.2</td>
<td>0.15</td>
<td>0.1</td>
<td>0.05</td>
<td>0</td>
</tr>
</tbody>
</table>

*Table 16: Panic values and the corresponding waiting radius*

The waiting times assigned are larger for traffic in the same direction (which usually corresponds to line) than for opposite traffic (which happens when there is not enough room for avoidance, and therefore agents may appear face to face until space is available to maneuver)
For animations showing “normal” situations (e.g., individuals leaving a cinema after a movie) all the agents exhibit waiting behavior when there is no available space ahead of them. The emergent behavior observed corresponds to queuing. Since agents use tangential forces to move within a crowd while avoiding others, the strength of those tangential forces will lead to narrow or wide queues, as can be observed in Figure 37.

In contrast to queuing behavior where all agents are facing in the same direction, stopping face-to-face is unlikely to happen unless the crowd is so dense that the tangential forces cannot modify agent trajectories. In this situation, agents will temporally wait until movement is possible again.

Figure 37: Examples of wide and thin queues emerging when animating a “normal” scenario
Animation Results on queuing behavior

As we have previously seen, natural queuing behavior arises from the combination of waiting behavior and strength or weights assigned to tangential forces used to avoid other agents moving in our same direction. When simulating a crowd under normal conditions (i.e., individuals will calmly wait for their turn to get through a door) autonomous agents use geometrical rules to alter the forces model and make themselves stop until the geometrical space defined has cleared up. At the same time, as agents move through a crowd, tangential forces are used to walk around other individuals. The intensity of these tangential forces is what allows us to achieve different queuing styles as can be observed in Figure 38.

![Figure 38: Different emergent queuing behavior based on the weights of the tangential forces. Weights from top-left to bottom-right are 0.1, 0.5, 2.0, 5.0](image)
Figure 38 (a) shows line formation for a very small weight of the tangential force (0.1), the result is an almost perfect line of agents waiting one after the other. The Figure 38 (b) corresponds to a slightly higher weight (0.5); the emergent line formation is still quite thin but we can already appreciate the difference from Figure 38 (a). Figure 38 (c) and (d) show increasing values of the weight (2 and 5) and we can observe how as the weight increases, the emergent waiting behavior turns into wider groups or lines of agents.

5.5.5. Pushing behavior

Pushing behavior emerges since HiDAC can handle not only collision avoidance but also collision detection and response. Agents have different behaviors that can be triggered at any time. During an organized situation, individuals wait for space available before moving, but when in panic, they try to move forwards until they collide with other individuals and therefore no progress can be made. By combining both behaviors simultaneously for a heterogeneous crowd, we achieve an emergent behavior where some individuals that do not respect personal distances will get very close to other agents and be able to push them away in order to open their way through a dense crowd. The effect of being pushed away is achieved by applying collision response forces.

Figure 39: Pushing forces
Figure 39 shows a sequence of three times $t_0 \ t_1 \ t_2$ where pushing forces are applied. At time $t_0$ agent $A_0$ is moving with desired velocity $v_0$ towards agent $A_1$ who has desired velocity $v_1$. Personal space is represented by a dotted line around the agent’s body. As we can observe in the Figure, $A_0$ has a smaller threshold for personal space than $A_1$. At the next time step $t_1$, $A_0$ manages to move within the personal space of $A_1$ and creates a repulsion force $r_{01}$ that will force $A_1$ to be pushed away. Since repulsion forces apply when an agent intersects another’s personal space, there is no repulsion force affecting $A_0$ since its personal space is not being intersected. In the next time step $t_2$ we can observe how $A_1$ has been pushed away.

The desired velocity vector for $A_1$ is not affected given the fact that the velocity vector only changes when the agent decides to steer (avoiding obstacles, moving towards attractors, slowing down, etc.). In this case, $A_1$ has not made a movement based on its decision of changing trajectory but on being pushed away, and therefore $A_1$ continues with the previous desired velocity.

The user can specify the panic parameter through the interface by setting the percentage of people that will exhibit panic behavior as soon as an alarm goes off.

Figure 40 gives an example where the top left room has been filled with panicked people (represented by red-heads) who will tend to push others away, while the other three rooms contain individuals following more organized behaviors. After some seconds of animation, the red-headed people have managed to almost empty their room while pushing others away in the corridor in order to reach the exit faster. Individuals in the other rooms are calmly waiting for their turn to get through the door.
Figure 40: Red-headed people exhibit panic behavior and push others to open their way through the crowd.

**Animation Results on pushing behavior**

The pushing behavior makes a significant difference in the types of emergent behaviors that can be achieved compared to other methods for crowd animation. Such methods do not apply collision response and consequently high density crowds turn out to either have agents overlapping or to always exhibit stopping behavior so as not to overlap with each other.

Pushing behavior allows for more realistic high density crowds, where agents are pushed off course due to a large number of people moving in a different direction, agents push people in a line to get through the crowd faster, or even reach the extreme scenario of pushing other agents to the floor.

Figure 41 shows an emergent maneuver where an agent (with red head and white shirt) is trying to move in the direction indicated by the black arrow, but instead makes an arched trajectory as indicated by the yellow line, due to the fact that there is a dense crowd trying to move in a different direction and consequently pushing any people on their way.
5.5.6. Falling and becoming obstacles

When the majority of pushing forces affecting one individual are approximately in the same direction, the agent will receive a sum of forces with magnitude high enough to make it lose equilibrium. At this moment the person may fall and become an obstacle for the rest of the crowd.

Fallen agents represent a different type of obstacle because, unlike walls and columns, a body on the floor is an obstacle that should be avoided, but if necessary (or unavoidable) one can walk over it. Thus, it needs to be treated differently. In HiDAC, fallen individuals become a rectangular obstacle (a bounding box covering the torso and head, but not the legs since other individuals can easily walk over that part of the agent). When other agents approach this new obstacle, weak avoidance forces are applied in order to walk around the fallen agent. However, repulsion forces are not applied. Therefore, when the crowd is extremely dense and the pushing forces from behind are strong, the result is that agents may walk over the body on the floor, as has been observed in actual extreme situations. Figure 42 shows an example of this type of obstacle avoidance behavior when the density of the crowd is artificially low.
There are two possible types of avoidance forces. The first one correspond to the case where the agent is within a neighboring area of the rectangle bounding the fallen agent, and it is calculated as tangential to the closest side of the rectangle. The second type corresponds to the case where the agent is already stepping within the rectangle. In this case, avoidance forces are a combination of tangential forces and the normal of the side that is closer to the agent. These two types of avoidance forces can be observed in Figure 43 respectively.
The user can specify the percentage of the population that has a high probability of falling through the parameter *falling* from Table 7. For that segment of the population, agents will fall if the sum of repulsion forces affecting them has magnitude higher than 0.1 and a random number obtained in the range [0,99] is higher than 30, which means they have 70% probability of falling when being pushed.

**Animation results on falling and becoming new obstacles**

In the next two animation examples we can observe the avoidance behavior of the crowd depending on the density.

In the first example (Figure 44) the crowd is relatively dispersed so the avoidance forces that the fallen agent exerts on the other agents are enough for them to modify their trajectory and walk around it. In the Figure we have tracked the trajectory of one of the agents (marked with a red circle) in order to see the final path followed around the fallen agent.

![Figure 44: agents avoiding fallen individuals.](image)

In the second example (Figure 45) we can observe a high density crowd, where even though the agents try to walk around the fallen individual, we see how some of them get pushed and cannot avoid stepping on or walking over the fallen individual. We tracked the path of one of the agents (marked with a red circle) to show how he gets pushed towards the fallen agent even though he is trying to move around it.
5.5.7. Panic Propagation

Panic is a very important feature when simulating evacuation in crowded environments. In HiDAC it is possible to model the effects of panic not only in the high level decision making process, but also in the low level behavior of the agents. As observed in the psychology literature, one of the most distinguishing features of people under panic is the fact that they will try to do everything faster, and thus speed up. This behavior is easily exhibited by increasing the speed of those agents that tend to panic when an alarm goes off. In our system, panic will not only change agent’s speed, but will also affect some low level behaviors such as canceling out the waiting behavior and activating instead the pushing behavior while reducing personal space thresholds. We explained in section 5.5.1.1 how smaller personal distance thresholds on an agent yield stronger repulsion forces on the agents surrounding him. Therefore the agent exhibiting panic behavior can push others in order to open a path through the crowd.

A more interesting effect to simulate is how panic can be spread among a crowd. Some individuals that will normally not tend to panic during an emergency may exhibit more agitated behavior if they find themselves within a panicking crowd with people pushing and creating a claustrophobic effect by leaving little or no space to move in. This effect is also modeled in HiDAC. When an alarm goes off some agents will start in the panic behavior mode. In this mode they tend to push, exhibit agitated
behavior and some have the ability to avoid bottlenecks. All these behaviors depend on the agent personality and levels of panic. As the agents start running, they may provoke panic in other agents who will in turn modify their behavior. In order to propagate panic, we use either communication between agents (managed by the high-level behavior module), or perception to detect relevant changes in the scenario such as increasing crowd densities and number of people pushing (both low-level behaviors).

The percentage of agents that will propagate panic can be specified through the interface by assigning a value to the panic propagation parameter from Table 7.

In Figure 46 we can observe a sequence of images where the panic behavior gets gradually propagated among the crowd. We visually represent people in panic by using red-heads.

Figure 46: Panic propagation sequence
Chapter 6

Wayfinding with Communication and Roles

6.1. Introduction

In order to have agents moving within a dynamic complex environment we need to have a high level algorithm that will allow the agents to explore the environment and learn its features to perform the right navigation.

Agents move within complex virtual environments with several rooms, corridors, obstacles, stairwells and doors that can be opened or closed at any time during the simulation (Figure 47). In order to navigate this virtual environment, route selection is carried out through an interactive high level way-finding algorithm that dynamically calculates the global path based on the agent’s knowledge of the environment [PB06].
6.2. High level algorithm

Once the maze has been created and the cell information has been automatically generated, the crowd simulation algorithm proceeds through three main steps: (Figure 48).

I. The first step is the communication process. All the agents within a room share their knowledge about the environment (their mental maps containing information about blocked cells and subgraphs that have been fully explored finding no exit). At every time step we are computing a high level path over the cell-and-portal graph, which contains the information about the order in which the cells should be visited to get to an exit, and it is thus an ordered sequence of cell identifiers and portals connecting adjacent cells.
II. In the second step each agent will check if the known shortest path has no known hazards, and therefore it will just follow that path while adding the next cell to its mental map.

III. The third step occurs when there is some hazard in the shortest known path. In this situation, depending on the type of agent, there are different behaviors. If it is a trained agent, then its mental map contains the entire maze graph with all the portals, and therefore it can just follow the next shortest path known from its current cell. If it is an untrained agent, then it needs to explore the building in order to find new routes towards the exit. This exploration will be done through Depth First Search (DFS). Since initially the agent does not have the entire maze graph in its mental map, this DFS is implemented in an iterative way, so that it will discover new rooms only when it sees a portal and crosses it. For untrained agents, it could also be a “follower”, which means that if it does not know what to do, and as long as it can see another agent in the room, it will tend to follow the decisions taken by the other person instead of doing a DFS of its own.

Figure 48: High Level Wayfinding diagram
6.2.1. Exploring the building

When the known paths appear to be blocked, each agent needs to perform a graph search in order to find its way towards an exit. Those agents that are not trained will lack complete information of the environment, and therefore traditional global path planning techniques such as A* cannot be applied. The exploration algorithm used is based on a DFS algorithm [Kwe97]. At any given time, the mental map that an agent has its internal memory will be a connected subgraph of the cell and portal graph that represents the environment. As an agent walks towards the exit, each cell visited will be added to its internal mental map (memory) and marked as \textit{VISITED}. Initially the agent knows all the traversed edges (portals) and visited rooms (cells) but ignores how many rooms the environment has and where each portal that has not been traversed leads to. When the agent finds its desired path blocked, it then needs to search for an alternative path. The agent will start exploring those rooms that appear as neither \textit{VISITED} nor \textit{FINISHED}. \textit{FINISHED} cells are those that have been visited and either there are no more portals leading to other cells, or all the adjacent cells have also been marked as \textit{FINISHED}. If from the current room all the adjacent rooms have been either \textit{VISITED} or \textit{FINISHED}, the agent will chose a visited one randomly, since unlike \textit{FINISHED} cells, visited cells can still lead to new paths.

The following diagram shows very briefly an iteration of the algorithm:
The main difference between DFS in graphs and what our agents perform is that in graph search, DFS is performed in a sequential manner going through all the cells. Our agents can benefit from the communication process with other agents, which will allow them to prune their graph search. When two or more agents meet within a room, they can exchange information about hazards and parts of the building (subgraphs) that have been fully explored by others, where no exit was found. The mental maps can be updated after visiting a room or after communication with other agents.

In order to make the agents’ behavior closer to real humans, we need to have some considerations in mind. For example:

- People during a conversation are unable to give such detailed information in terms of room connectivity about big subgraphs. Therefore the information is limited to two levels of adjacency from the current cell in the cell-and-
portal graph representing the virtual environment. We can think of it as: e.g., “the door on the left leads nowhere”, “the room on the right leads to another office, where there’s no exit either”).

- People in panic tend to get disoriented. Therefore when an agent is in panic, part or all of its internal memory, could be “forgotten”.
- People in panic may also change their role from leader to follower. Therefore an agent that was performing a search, after being affected by the panic behavior, may start following others instead of performing its own search.
- When dealing with dynamic environments (e.g., portals that are lock or unlock at different times) agents may have explored the entire graph, but if no exit has been found yet, then they will keep on searching hoping for a door to get unlocked.

6.3. Results on Wayfinding and Decision-Making

Our goal is to study the performance of different search algorithms when large groups of agents with individualistic personalities use communication in order to reduce their graph search space, and then show how psychological elements can alter the high level decision making process to achieve more realistic behaviors.

The experimental results are divided in the following ways:

- Comparison of random search vs. depth first search
- Comparison of communication vs. non communication
- Comparison of trained leaders vs. untrained leaders
- Importance of leadership
- Psychology affecting roles and navigation
- Interactively modifying navigation based on impatience and changes in the environment.
The random search has been implemented exclusively for benchmarking purposes, since it is not the most realistic behavior for humans. In the second comparison, we can appreciate the significant impact that communication has in the behavior of the crowd when executing wayfinding. Finally, we will show the impact of having trained agents in the crowd and we will analyze the percentage of leaders that is actually useful to speed up the evacuation process.

For the experiments, we use 3 different scenarios, all of them maze-like. Two of them have been randomly generated, and the third has been created by hand in order to produce a maze that better resembles a real building. The three mazes contain 100 rooms, 8 of them blocked by some hazard such as fire. For each set of parameters, we have run 25 randomly generated starting configurations for the crowds.

The populations used for these results go from N=20 to 200 agents. The levels of leadership go from 0% to 100%. No leaders means they are all followers, and therefore when several agents meet in a cell, one random agent makes a decision and the others will just follow. This case implies dependent agents: when they find themselves in a panic situation they will always follow other agents instead of making their own decision. On the other hand, 100% leadership means each of them will perform its own decision making process with its current knowledge, which in the case of trained people should be complete knowledge of the building’s internal structure.

6.3.1. Comparison of random search vs. Depth First Search

In order to explore the building once all the known shortest paths are found to be blocked, we implemented two different algorithms. The first one represents a naïve search, where individuals start exploring randomly adjacent rooms trying not to go backwards unless they find themselves trapped. In this naïve search agents lack a mental map of the model nor can they create one while navigating the environment, therefore the emergent behavior obtained looks quite chaotic. On the other hand, a
DFS algorithm was used to make the agents search adjacent rooms in a more structured way while they create their mental maps. The results obtained showed not only that DFS was about 15 times faster than random search, but also the emergent behavior obtained was closer to the behavior expected of a real crowd, since the paths followed are more coherent and do not look as chaotic as in random search.

6.3.2. Comparison of communication vs. non communication

In Figure 50 we can appreciate the different performance of the algorithm with and without communication for 200 agents. We are interested in finding the simulation step at which the simulation converges to 100%, meaning that the entire crowd has evacuated the building.

As Figure 50 shows, the simulation with communication converges to 100% in almost half of the time that it takes the non-communication case to converge.

Figure 51 shows the results obtained for different crowd sizes where all the agents represent independent individuals (they will make their own decision during wayfinding instead of following others). In this simulation we are not yet considering
trained agents, therefore all the individuals in the crowd are unfamiliar with the internal structure of the building, and will find out how to evacuate the building based on their own exploration of the building and the shared communication. The plot shows the evacuation times for crowds of size: 20, 60, 100, 150 and 200 agents.

![Evacuation times for varying population sizes](image)

*Figure 51: Evacuation time for different crowd sizes using communication. 100% untrained leaders*

As we can see, the evacuation time decreases as the crowd size increases. This can be explained by the fact that for bigger crowds, the probability of meeting another agent increases, and therefore the important information about hazards in the building and explored areas spreads faster among the individuals. This information helps agents to prune their graph search and therefore find the correct path sooner.

It is important to notice though, that this holds as long as the crowd is not so large that congestion blocks the doors, which will obviously decrease the evacuation time. This problem can be observed for crowds of over 500 agents, where the evacuation time is constrained by the number of exits and the flow rate through each of the doors (Figure 52).
6.3.3. **Comparison of trained leaders vs. untrained leaders**

We performed 25 simulations using a crowd size of 100 and 0, 25, 50, 75 and 100 percent of trained agents. Figure 53 shows the average evacuation times obtained.

![Evacuation times for different percentages of trained leaders](image)

*Figure 53: Evacuation time for 0, 25, 50, 75 and 100% leadership*
As expected, we observe that the percentage of evacuated people converges to 100% faster as the percentage of trained people increases. This seems an obvious result to obtain given that trained people know how to evacuate a dangerous location because they have more information about the environment and dependent agents will follow them, therefore the overall evacuation time will decrease as the number of trained agents in the environment increases.

Not everyone needs to be trained, however. We can determine what is an adequate percentage of leaders needed to have a fast evacuation. We have previously observed that there is not a big difference in the convergence values between 50% and 100% leadership, which means that there is no need to have a great proportion of leaders. Figure 54 shows smaller percentages of leaders:

![Evacuation times for different percentages of trained leaders](image)

*Figure 54: Evacuation times for small percentages of trained leaders*
Here we can conclude that an optimal percentage of trained people during the evacuation would be only around 10%. For lower values the evacuation time for the same percentage of evacuees is at least doubled. On the other hand having more than 10% trained people would only provide an evacuation speedup at most 16%.

6.3.4. Importance of leadership

We have shown the behavior of trained leaders against dependent agents with a “follow the leader” behavior, but there is still another case to consider. In real life some people have a higher probability of becoming leaders for a group of people when an emergency occurs. They are usually independent individuals that by nature are able to handle emergency situations better and also tend to help others. In MACES, these people correspond to untrained leaders.

We show two images of the evacuation process. The first one (Figure 55) corresponds to a population with a high percentage of leaders, where most of the individuals in the crowd tend to make their own decisions when attempting to exit the building. The second sequence (Figure 56) corresponds to a population with a high percentage of dependent people, who will tend to follow any leader instead of deciding routes by themselves.

In the first population we can observe an emergent behavior with lots of small groups of people:
In the second population, the emergent behavior shows fewer but larger groups of individuals. When the number of dependent individuals is higher, the size of the groups formed tends to increase, since dependent people will not leave a group to try to explore new paths on their own. Instead they tend to stay together and just go where the leader decides to go.
6.3.5. Psychology affecting roles and navigation

As indicated by the psychology literature, people under panic may reach the point where they are unable of making their own decisions. Most people react to time pressure through an increase in the speed of their actions, as well as by subjectively filtering information. In general, the evacuation of a building due to imminent danger is accompanied by considerable physical and psychological stress. Since rising stress levels have the effect of diminishing the full functioning of one’s senses, this leads to a general reduction of awareness, especially the ability to orient oneself quickly in rooms and surrounding areas [WSG03].

These behaviors need to be modeled in crowd simulation in order to improve the realism of the output. In MACES+HiDAC, stress due to imminent danger is simulated through panic. People under panic can change their role, going from leader to dependent individual. Therefore as panic spreads, more people will become dependent and the overall evacuation time will change since dependent individuals do not contribute in terms of exploring and sharing information to speed up the wayfinding process of the group as a whole.

It is also necessary to influence the agents’ ability to orient themselves quickly within the virtual environment. When individuals are under high levels of stress due to panic, their memory will be affected. We alter the mental map of the environment, by removing sub-graph of their internal memory, and therefore these agents may exhibit disorientation by walking again through an area of the building that they had already visited instead of moving towards unexplored areas of the building.

In Figure 57 we can observe the high level navigation performed by one agent under normal conditions. Initially the agent only has knowledge about one short path to each of the two exits in the environment. Initially he tried to reach the North exit, but after finding that door closed, he starts walking down the corridor towards the South exit. The agent has no knowledge about the alternative route through the south-
west room, so when a door on his known South path blocks him, he will start to explore the environment as indicated by the white-dotted path.

Figure 57: Autonomous agent exploring a virtual environment under normal conditions
The agent will not walk again into a room that has already been visited and where no exit or alternative doors were found.

From the psychology literature we know that a person in panic is very likely to feel disoriented, which can be modeled through memory decay based on the level of panic associated with the agent. We repeat the example above, but this time use an agent suffering from high levels of panic. Figure 58 shows the resulting trajectory. At the beginning the agent follows the same trajectory as under normal conditions and tries to exit the building through the known shortest exits. Once the agent finds those known paths blocked, the exploration stage is initiated. Since this agent suffers from memory decay, he happens to walk several times in rooms that were previously explored which makes the overall evacuation time longer and the path followed more chaotic.

Figure 58 shows six snapshots at different times during a simulation. The path followed in the time period between each snapshot is shown with a white-dotted line. The path that goes from point 0 to 1 indicates the initial trajectory when the agent is following the known shortest paths and, after finding them blocked, then starts exploring the building in a way similar to the example without panic. From point 1 to 2 the agent explores and ends up in the same doorway. From 2 to 3 we observe how the agent ends back in the initial position after trying to explore a few other rooms on its way. Finally, we can observe how the agent moves from point 3 to 4 and finds its way out. The last image shows the entire path followed.
Figure 58: Autonomous agent exploring a virtual environment in panic and therefore exhibiting disorientation.
6.3.6. **Interactively modifying navigation and impatient agents avoiding bottlenecks.**

In any real-time animation of crowds it is important to demonstrate autonomous agents showing both reactive low level behavior that affects their local motion based on dynamic obstacles or agents, and reactive high level behavior based on real time changes in the environment. These changes can be instigated by environment modifications (such as a door being closed, or a hazard such as fire spreading) or congregations of agents generating bottlenecks. The autonomous agents in our system can rapidly react to these changes by detecting them and modifying their high level navigation to plan an alternative route more applicable to the current situation.

When dealing with high-density crowds in buildings, bottlenecks can appear in the portals. HiDAC incorporates a high-level decision process that will allow impatient agents to react to this situation by finding an alternative path. As the low-level algorithm detects the bottleneck, it sends that information to the high-level which will try to find an alternative route based on what the agent can perceive from its current position (doors, obstacles) and the knowledge that the agent has about the internal connectivity of the building. If an alternative path is available, the high-level chooses a new portal as the goal and sets an attractor point to change the direction of movement. Only impatient agents will exhibit this type of behavior. The user can set the percentage of impatient agents in the population through the parameter *impatience* from Table 7

Changes due to congestion are driven by the impatience attribute and environment knowledge. Agents will choose alternative routes depending on their personal impatience value, the current congestion at the destination door, the distance left to reach that particular door and the known alternatives. Figure 59 gives six snapshots of an animation where impatient individuals (represented by red-heads) will perform a new high level planning in order to choose an alternative route that allows them to detour around the congested area.
When a change occurs in the environment (e.g., a door is blocked) agents perceive and react to it. For a door change, the high-level algorithm needs to make a new wayfinding decision. This decision is sent to the low-level motion by modifying the next attractor point towards which the agent needs to walk. The agent detects this change in real time and, as the new attractor becomes available, the agent needs to steer towards the new destination point.

Figure 60 shows an animated sequence where dynamic wayfinding is exhibited. Image (a) shows the initial configuration consisting of a simple building with one corridor leading to the exit and four rooms connected to the main corridor. The bottom right room also has an exit door. Initially suppose that the agents in each of the rooms know about the corridor exit. Only those agents in the bottom right room can perceive there is another exit. When the simulation starts, agents walk towards the corridor except those in the bottom right room who walk towards their closest
exit. Figure 60b shows the moment in which the exit in that room gets locked. Agents in this room immediately start moving towards the corridor to get to the next available exit, Figure 60c. As the agents walk down the corridor, the second exit also gets locked (Figure 60d). At this moment, the agents do not know how to get out of the building. The high-level module will have to perform navigation to explore and learn about the environment, while setting new attractors to animate the agents. For some time, individuals wander around the building looking through doors trying to find a solution (assuming they didn’t previously know where the other rooms lead to). Suddenly the door in the bottom right corner is opened again (Figure 60e). Agents do not know about this until they walk through the interior door and perceive it (Figure 60f). Eventually all the agents manage to leave the building. This example shows the interaction between high-level and low-level to achieve realistic simulations with dynamic changes in the environment geometry.

This interactive planning behavior will be exhibited by those agents that have the ability to explore the environment; those with a leadership role or those with a dependent role who find themselves isolated.
Figure 60: Crowd reaction to dynamic changes in the environment.
Chapter 7

Summary of Results

To test our crowd simulation system (HiDAC+MACES), we developed a building editor to rapidly create new virtual buildings to use in our experiments. Buildings of many layouts and floors can be created. Floors are linked through stairwells, and obstacles such as columns can be placed anywhere within the environment. Once the buildings are generated, we automatically create the cell-and-portal graph as explained in section 3.1 to calculate and store all the required information to be used during the simulation of the autonomous agents’ movements. The building editor is explained in detail in Appendix A.

Our crowd simulation system (HiDAC+MACES) has been developed in C++ and OpenGL, and is designed to run large crowd simulation scenarios. This system allows us to study both the wayfinding behavior of autonomous agents as they communicate and learn the features of the environment, and local motion behavior such as velocity, direction, and emergent queuing and pushing behavior. We also show results of how psychological elements can be employed to achieve highly realistic behavior close to that reported in the literature.

All the different parameters affecting agent behavior can be tuned by the user in order to simulate different types of crowds depending on the population that needs to
be modeled. In chapter 6, we showed the results achieved for high level wayfinding and how the emergent behavior depends on communication and roles. We also showed how psychological elements can be used to modify both wayfinding and local motion behavior.

Table 17 summarizes the options considered and results achieved with MACES.

<table>
<thead>
<tr>
<th>Options Assessed</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random search vs. DFS</td>
<td>DFS is 15 times faster than random search. Random exploration was too chaotic and therefore did not look realistic. DFS better simulates the expected approach that real people would take to navigation.</td>
</tr>
<tr>
<td>Communication vs. non-communication</td>
<td>The use of communication halves the overall evacuation time, since agents can exchange relevant information regarding evacuation routes and hazards.</td>
</tr>
<tr>
<td>Trained vs. non-trained leaders</td>
<td>As expected, trained people (building knowledgeable) evacuate faster. However, from our experiments, it turns out that not everybody needs to be trained, it is sufficient to have only 10% of trained leaders to yield results close to those where everybody is trained.</td>
</tr>
<tr>
<td>Impact of leadership</td>
<td>Having a crowd with a high percentage of leaders yields lots of small groups of people exploring the environment together. On the other hand, having a large percentage of dependent people yields fewer but larger groups of agents.</td>
</tr>
<tr>
<td>Psychology affecting navigation (and roles)</td>
<td>High levels of panic increases the probability of people switching from leader role to dependent role. Panic also affects the orientation skills of the agents, making them less able to remember the geometry and therefore leading to longer evacuation times and more chaotic-like navigation.</td>
</tr>
<tr>
<td>Interactive navigation</td>
<td>Changes in the environment are perceived in real-time by the agents who can modify their desired paths based on the current state of the world. Psychological factors, e.g. impatience, lead to modification of high level decisions to avoid bottlenecks.</td>
</tr>
</tbody>
</table>

*Table 17: High level results summary*
In chapter 5, we presented a number of representative simulations that show the visual output of HiDAC. In this section we summarize the results achieved through our local model, HiDAC. More videos showing all the different results can be found at: http://www.seas.upenn.edu/~npelecha/MACES/MACES.htm

Specifically, we have presented methods of achieving the following goals that enable realistic simulation of high-density crowds.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast perception of environment</td>
<td>Fast perception method (linear cost) that proves to be sufficient for local movement</td>
</tr>
<tr>
<td>Eliminate shaking behavior</td>
<td>More realistic movement for high density crowds, where agents do not appear to vibrate continuously.</td>
</tr>
<tr>
<td>Natural bi-directional flow and overtaking</td>
<td>Natural counter-flows for different density crowds. Also, overtaking behavior exhibited when agents with different velocities walk in the same direction and there is not immediate oncoming traffic.</td>
</tr>
<tr>
<td>Emergent queuing behavior</td>
<td>Lines emerge naturally when large groups of agents try to walk through a door. Line width varies in accordance with the tangential weights applied.</td>
</tr>
<tr>
<td>Realistic pushing behavior</td>
<td>Agents with smaller pushing thresholds are more likely to push others away when they are in panic. Pushing behavior emerges when groups of agents are pushed off course by the crowd.</td>
</tr>
<tr>
<td>Falling agents becoming new obstacles</td>
<td>Variety of emergent behaviors where agents avoid fallen victims by walking around them when the density of the crowd is low. When the density is high, agents may be pushed over the fallen victim.</td>
</tr>
<tr>
<td>Panic propagation</td>
<td>Panic can appear to spread through the crowd based on the panic propagation parameter. Agents in panic will exhibit faster movements and increased willingness to push others if necessary.</td>
</tr>
</tbody>
</table>

*Table 18: Goals and results for low level motion*
7.1. Performance

We have run several simulation tests on a 2.99 GHz Intel Xeon with 2GB of RAM measuring frame rates both for simulation only, and for simulation and 3D rendering. When doing simulation only, our system can handle up to 1800 agents with a frame rate of 25Hz. Simulation and 3D rendering using an NVIDIA Quadro FX 3400/4400 graphics system can achieve 25 frames/second (not using GPU rendering) for up to 650 simple 3D virtual human figures each with about 100 vertices. For the frame rate tests, we used a large complex environment with 85 rooms and 53,448 vertices overall.

Figure 61 shows the frame rate achieved when rendering the 3D geometry in real time for a population of agents ranging in size from 100 to 1000.

![Frame Rate for 3D Rendering](image)

*Figure 61: Frame rate for 3D rendering vs. agent population*

Figure 62 shows the time in milliseconds that takes to calculate and render each frame depending on the population size.
Figure 62: Milliseconds per frame vs. population size

Figure 63 shows the frame rate achieved when performing simulation only for a population of agents ranging in size from 300 to 1800. Finally, Figure 64 shows the time in milliseconds that it takes to calculate each simulation step depending on the population size.

Figure 63: Frame rate vs. population size for simulation only
Figure 64: Milliseconds per Frame vs population size for simulation only
Chapter 8

Conclusions

We presented a framework to realistically simulate high density crowds affected by psychological and physiological elements within complex virtual environments. The system combines two subsystems: MACES and HiDAC. MACES deals with high level decision-making and navigation using communication, roles and psychological influences. HiDAC deals with the local motion of the agents to achieve believable agent movement for high density crowds and delivers a wide variety of emergent behaviors driven by the psychological and physiological capabilities of each agent.

MACES implements high level wayfinding during an evacuation scenario for crowds unfamiliar with the internal structure of the environment. We included individualism in the agents by assigning them different roles (trained leaders, untrained leaders, and followers). The flexibility of the model allows for variations in the number of people, building structure, number of hazards, and combinations of roles for the agents.

The main contributions of MACES are:

a) A high level wayfinding algorithm to allow individuals in a crowd to explore an unfamiliar building in order to find exits during an emergency.
b) Using inter-agent communication to share knowledge of the building during high level wayfinding.

c) Inclusion of roles to provide individualism into the crowd. Agents have a given personality that will drive high level behavior, while they are also endowed with psychological elements such as impatience and panic that can affect its internal state at any time and consequently modify its overall behavior.

d) Inclusion of psychological elements (panic and impatience) that affect agents’ wayfinding by introducing orientation difficulties and interactive path planning based on changes in the environment and bottlenecks.

Our evaluation has shown a significant improvement in crowd evacuation rates when using inter-agent communication. Also we can observe the grouping behavior that emerges when there are a high percentage of dependent agents in the crowd. Only a relatively small percentage of trained leaders yields evacuation rates comparable with 100% trained leaders. We have also shown how psychological factors can affect agents’ wayfinding in terms of impatient individuals being able to detect bottlenecks and modify their paths based on their current knowledge, and agents in panic becoming disorientated and either walking several times through previously explored areas or becoming dependent agents unable to explore the building by themselves.

HiDAC can be tuned to simulate different types of crowds, ranging from extreme panic situations (fire evacuation) to high-density crowds under calm conditions (leaving a cinema after a movie). The system has been calibrated using data from real human behavior in order to exhibit reasonable velocities, flow rates and densities.

Unlike cellular automata and rule-based models, HiDAC can realistically simulate an individual trying to force its way through a crowd by pushing others, and unlike social forces models, our agents can exhibit more respectful behavior when
desired and take decisions in terms of letting others walk first, queuing when necessary and waiting in line. These emergent behaviors are driven by the combination of psychological and sociological rules that we layer on top of our extended social forces model.

A fast perception method has also been implemented that allows for realistic obstacle avoidance and emergent bi-directional flow rates. Impatience has been integrated in the high level module in order to avoid the sheep-like behavior that many other crowd simulation models exhibit.

We have shown novel extensions to the social forces model by adding braking forces and influence region controls that mitigate agent vibration. For local motion simulation our system uses the best features of both rule-based and social forces systems, while eliminating many of their disadvantages. This implementation allows real-time simulation of many hundreds of agents.

Our contributions at the local motion level, driven by psychological and physiological parameters, are:

a) Eliminating shaking behavior implicit in the basic social forces model. The method consists of applying braking forces to the social forces model when repulsion forces coming from other agents appear opposite to the desired direction of movement.

b) Fast perception method (linear cost on the number of obstacles per room) unlike ray-casting which has cost $O(N \cdot R)$, where $N$ is the number of obstacles, and $R$ the number of rays casted. Our method is based on having influence rectangles and prioritizing obstacles based on distances, angles and directions of movement.

c) Natural bi-directional flows and overtaking based on a combination of variable length rectangles of influence, differential right preferences, and relative direction between autonomous agents.

d) Emergent queuing behavior by using influence discs that trigger waiting behavior based on agent direction. This, combined with different
tangential weights for the avoidance forces, yields a variety of line/queue formations.

e) Realistic pushing behavior achieved by applying collision response based on pushing thresholds / personal distances.

f) Falling agents becoming new obstacles. These obstacles are addressed by applying weak tangential forces (but not repulsion forces), which does not guarantee that the agents will always walk around the fallen victim.

g) Panic behavior and panic propagation: Panic does not only serve to increase the velocity of the agents. In HiDAC, panic affects velocities and overall behavior by driving agents to not respect lines and by modifying pushing thresholds. Panic behavior can be perceived by other agents in the crowd (and given their personality parameters), who may also start exhibiting that type of emergent behavior.

The framework presented in this dissertation combines high-level decision making including communication and roles (MACES), with a low-level local motion system (HiDAC) that can exhibit a large variety of emergent behaviors. The two systems interact in real-time while being driven by a set of psychological and physiological parameters that allow the user to have control over the final behavior exhibited by the crowd.
Limitations and Future Work Plan

Despite our improvements to existing systems, there is still a large amount of work to be done to bridge the gap between crowds of real humans and crowds based on multi-agent systems. Some of the enhancements that could be incorporated with our current framework include:

- Integrate our system with a more complex cognitive model that could modify the agent’s state to achieve an even wider variety of behaviors.
- Improve the graphical forms of the very simple 3D figures that were used to represent the agents. Although they are sufficient to visualize the results, the world of computer graphics demands that we incorporate more sophisticated, fully articulated and textured human figures that would greatly improve the overall visual appearance of the animation.
- Integrate better model input capabilities. Although our building editor was sufficient to rapidly generate large complex environments, in the future we should to be able to load CAD geometry into our system. For this purpose, we would need to generate the cell and portal graphs from CAD files.
- A depth first search algorithm has been implemented to simulate human navigation for indoor scenarios. We would like to run VR experiments with real people to study human navigation and modify the current wayfinding algorithm to achieve results closer to actual human performance.
- In our system memory decay is affected only by time. It is also important to consider that other factors such as building geometry play an important role in someone’s ability to remember internal features [NCK05]. Therefore, further studies should be done in order to implement a memory decay function that can simulate the way real humans remember or forget things. This could be studied within the context of the VR experiment.
- Finally, we have demonstrated the importance of incorporating communication into the simulation of large numbers of individuals. Real
humans interact with each other in many ways, and therefore it is essential to incorporate these into the simulation if we want to achieve realistic results. Our current system, which uses limited forms of verbal (information exchange) and non-verbal (perceiving panic) communication, has shown results where the overall emergent behavior is close to the expected one. However, there are still many other elements that should be communicated, and that can also be used for coordination among different members of the crowd. For example, our agents can help each other in the wayfinding process by exchanging relevant information about hazards and explored parts of the building when they meet with others by chance. In the future it will be interesting to have the autonomous agents being able to coordinate themselves in terms of deciding which part of the building each one should explore and then working out how to meet up again to decide upon the best alternative.
References


[Che99] Cherif M.: Motion planning for all-terrain vehicles: a physical modeling approach for coping with dynamic and contact interaction constraints. IEEE transactions on Robotics and Automation (ICRA) 1999


[PM71] Predtetschenski W. M., Milinski A. I.: Personenstrome in Gebauden-Berechnungsmethoden fur die Projektierung (Muller, Koln-Braunsfeld, 1971).


APPENDIX
To facilitate rapid generation of a wide variety of large complex environments in which we could run our crowd simulation model, we implemented a building editor (Figure 65).

This editor allows us to create buildings of any number of floors while we can simultaneously visualize them in both 2D and 3D. The 2D view allows visualizing one floor at a time and specifying the geometry of the layout by locating walls, columns, stairs, windows, etc. just by clicking with the mouse in the position where we want to place the element selected from the right panel.

Once the building is created we can save it in ASCII format that can be loaded afterwards in our crowd simulation system in order to generate the cell and portal graph in which the autonomous agents can navigate. The ASCII format used allows
to easily picture the final layout and also straightforwardly add changes to the geometry even without needing to employ the editor.

Although this building editor is very limited in the type of buildings that we can create, it was out of the scope of this work to develop a complete building editor as the commercial software tools available, but instead, our goal was to have our own fast and simple way of creating new buildings complex enough to test all the abilities of the crowd simulation system presented in this dissertation. In the future we would like to load CAD files instead and create the cell and portal graphs from those files that are necessary for our crowd simulation system to animate the autonomous agents.

Figure 66: Example of high rise building editing and some 3D views.