Universitat Autònoma de Barcelona

Crowd Modeling and Simulation on High Performance Architectures

Albert Gutiérrez Millà

Universitat Autònoma
de Barcelona

Crowd Modeling and Simulation on High Performance Architectures

A DISSERTATION PRESENTED
BY
ALBERT GUTIÉRREZ MILLÀ
TO
THE COMPUTER ARCHITECTURE AND OPERATING SYSTEMS DEPARTMENT
UNDER THE SUPERVISION OF DR. REMO SUPPI BOLDRITO

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
COMPUTER SCIENCE

UNIVERSITAT AUTÒNOMA DE BARCELONA
CERDANYOLA DEL VALLÈS, BARCELONA
JULY 2016

Crowd Modeling and Simulation on High Performance Architectures

A thesis submitted by Albert Gutiérrez Millà in partial fulfilment of the requirements for the degree of Doctor of Philosophy, in the subject of Computer Science, under the supervision of dr. Remo Suppi Boldrito at the Computer Architecture and Operating Systems department.

Cerdanyola del Vallès, 2016:

Author: Albert Gutiérrez Millà

Supervisor: Remo Suppi Boldrito

Abstract

Management of security in major events has become crucial in an increasingly populated world. Disasters have incremented in crowd events over the last hundred years and therefore the safety management of the attendees has become a key issue. To understand and assess the risks involved in these situations, models and simulators that allow understand the situation and make decisions accordingly are necessary. But crowd simulation has high computational requirements when we consider thousands of people. Moreover, the same initial situation can vary on the results depending on the non deterministic behavior of the population; for this we also need a significant amount of statistical reliable simulations. In this thesis we have proposed crowd models and focused on providing a DSS (Decisions Support System). The proposed models can reproduce the complexity of agents, psychological factors, intelligence to find the exit and avoid obstacles or move through the crowd, and recreate internal events of the crowd in case of high pressures or densities. In order to model these aspects we use agent-based models and numerical methods. To focus on the applicability of the model we have developed a workflow that allows you to run in the Cloud DSS to simplify the complexity of the systems to the experts and only left to the them the configuration. Finally, to test the operation and to validate the simulator we used real scenarios and synthetic in order to evaluate the performance of the models.

Resum

La gestió de la seguretat en grans esdeveniments s'ha convertit en clau en un món cada vegada més poblat. Els desastres multitudinaris han augmentat en els últims cent anys, de manera que la gestió de la seguretat dels assistents és una qüestió clau. Per comprendre i avaluar els riscos involucrats en aquestes situacions els models i simuladors permeten comprendre la situació i prendre decisions en conseqüència. Però la simulació de multituds te alts requeriments computacionals si tenim en compte milers de persones. A més, la mateixa situació inicial pot donar resultat diferents degut a comportaments no deterministes de la població; per això necessitem una quantitat significativa de simulacions. En aquesta tesi s'han proposat models multitudinaris i ens hem centrat en crear un DSS (Sistema de Suport a les Decisions). Els models proposats poden reproduir algunes de les complexitats dels agents, factors psicològics, la intel·ligència per trobar la sortida, evitar obstacles o moure a través de la multitud, i reproduir esdeveniments interns de la multitud en cas d'altes pressions o densitats. Per modelar aquests aspectes fem servir models basats en agents i mètodes numèrics. Per centrar-nos en l'aplicabilitat del model hem desenvolupat un flux de treball que li permet executar el DSS en el núvol per simplificar la complexitat del sistema als experts. Finalment, per provar el funcionament i per validar el simulador es van utilitzar escenaris reals i sintètics que ens han permès avaluar el rendiment dels models.

Resumen

La gestión de la seguridad en grandes eventos se ha convertido en clave en un mundo cada vez más poblado. Las tragedias multitudinarias han aumentado en los últimos cien años, por lo que la gestión de la seguridad de los asistentes es una cuestión crucial. Para comprender y evaluar los riesgos involucrados en estas situaciones los modelos y simuladores permiten comprender la situación y tomar decisiones en consecuencia. Pero la simulación de multitudes tiene grandes requerimientos computacionales si tenemos en cuenta miles de personas. Además la misma situación inicial puede variar en sus resultados debido a los comportamientos no deterministas de la población; para esto necesitamos también una cantidad significativa de simulaciones. En esta tesis se han propuesto modelos de multitudes centrados en ser DSS (Sistema de Apoyo a las Decisiones). Los modelos propuestos pueden reproducir algunas de las complejidades de los agentes, factores psicológicos, la inteligencia para encontrar la salida, evitar obstáculos o moverse a través de la multitud, y reproducir eventos internos de la multitud en caso de altas presiones o densidades. Para modelar estos aspectos se utilizan modelos basados en agentes y métodos numéricos. Para centrarnos en la aplicabilidad del modelo hemos desarrollado un workflow que permite ejecutar el DSS en la nube para simplificar la complejidad del sistema a los expertos. Finalmente, para probar el funcionamiento y para validar el simulador se utilizaron escenarios reales y sintéticos que nos han permitido evaluar el rendimiento de los modelos.

Contents

Listing of figures

List of Tables

To all the people who asked me: "How is your thesis going?".

Acknowledgments

I first would like to thank my supervisor Remo Suppi for his support and for accepting me in the research group.

I want to express my gratitude to Zhengchun Liu and Dani Ruiz who have patiently helped me along this thesis, Claudio for his altruism and jokes and Francisco José da Silva (Paco), my research colleague and calistenia master.

Acknowledge professors Emilio Luque and Dolores Rexachs for their comments and contributions to this thesis.

Thank my colleagues for their support and their chats that have always been stress releasing: Alejandro, Arindam, Cecilia, César, Ferran, Hai, Javi, Joe, Jorge, Josefina, Laura, Marcela, Max and Pilar. Thank all the staff of the CAOS department, and also Fina who was always the first person to receive me every morning in the office and made me laugh with his jokes and honesty.

This research has been possible thanks to the MINECO (MICINN) under contracts TIN2011-24384 and TIN2014-53172-P and the PIF grant conferred by the Generelitat de Catalunya.

1

Introduction

In this chapter, we introduce our problem, crowd evacuations, as well as the problems that will be discussed along the thesis. The motivation to this topic is described as well as related issues. A proposed methodology is described and its parts are introduced. We propose a list of objectives for this thesis, and how we aim to contribute to the research field.

1.1 Motivation

Since the industrial revolution the world population has been growing in an remarkable way over the years. In Fig. 1.1.1 the population growth can be seen, and how it raised over six billion people the last two hundred years. Cities are rapidly increasing their population in such a fast and big manner that a new term has been invented for cities with more than 10 million people: mega-cities. The number has been increasing in the following years due to expansion of the populace and emigration from villages to big cities. Nowadays there are 35 mega-cities around the world. Therefore, crowds, which are a potentially dangerous situations, are getting more usual and they have become part of our reality. There is a need for the safety of the crowd and try to reduce any possible injury.

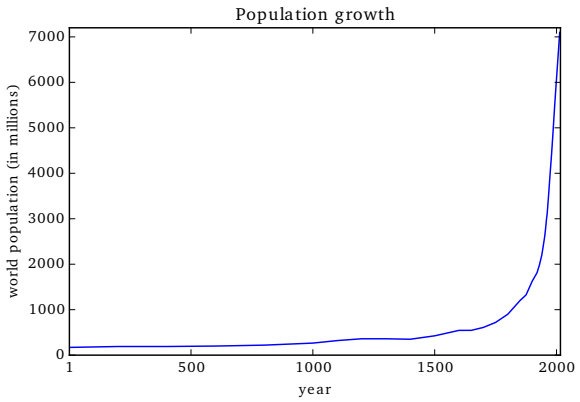

Figure 1.1.1: Population growth according to ourworldindata.org

In the last century, disasters in evacuation situations have increased. Cases as the Hillsborough Stadium disaster (Sheffield, England in 1989) causing 96 deaths, the Love Parade disaster (Duisburg, Germany in 2010) causing 21 deaths or the Madrid Arena disaster (Spain in 2012) leading to 5 deaths show us how bad decisions and planning can lead to human harm. These disasters have taught us that a good understanding and prevention is necessary in order to prevent and be prepared for any situation. We have to consider that disasters are unpredictable but we

need to know as much as possible and maximize the understanding of crowd situations and know what may happen to the assistants. When talking about crowd evacuations we will assume an order of hundreds or thousands of persons in a closed space.

Knowing the problematic areas, exits or conflict situations where the disaster may occur can be crucial in emergency planning. Moreover, understanding crowds is, and will be, a key factor of security and building and event design. Exercises such as fire drills have been an historical way to know the evacuation time of buildings and to train the population inside the building. But it is not possible to perform certain crowd drills where the people could be in a risky situation for their life. When coming to the reproduction of potentially dangerous situations such as high pressure crowds, or events with thousands of people this cannot be done. In the case of crowd evacuations, the complexity and uncertainty of the system increases. How to handle them and predict the behavior requires models, to understand and reproduce the behavior of the crowds. That way, the experts will have the knowledge to carry out what is thought appropriate in evacuation planning: the safest and fastest evacuation possible. In this thesis we work on this direction, proposing crowds models and contributing to the understanding of crowd events and phenomena through simulation.

There has been a historical human desire to understand the laws of the nature and model them. Modeling has allowed the humanity to advance its knowledge and improve its day living. In the particular case of evacuations and crowd dynamics there has been an increasing interest in the reproduction of evacuations through several approaches during the last years such as discrete time or continuous time models. These approaches differ on the focus of how to extract the natural characteristics of the reality and how to implement them. Each approach has pros and cons, and choosing one or another is always a trade-off.

Simulators implement the ideas designed by the models. Simulation allows us to interact with the model and generate results without interacting with the real system. This abstraction has been useful along history, to predict, explain the future, and also to reproduce and understand the past. This abstraction has been used

in weather forecasting, engineering design, drug design, evacuations, etc. Computer simulation allows us to run crowd dynamics models and extract information from these emergency situations. In our case models and simulator have mainly the purpose of provide knowledge of potential problems and also analyze theories and simulate them. Moreover, one of the main final goals is to provide a decision support system where the complexity of the model and the performance is hidden to the final user. Because we will be working with thousands of agents interacting between them and with the environment, and also reproducing different scenarios, the simulation will have high computational needs needing High Performance Computing (HPC).

When finding tools that describe the evacuation of a building or even an evacuation with fire, there are plenty of tools and models. But in the case where there are thousands of people there is more uncertainty and the number of people impacts on the unpredictability there are not simulators and there research is also pushing in this way. The different consequences from the same configuration and scenario, the psychological and sociological factors or the size of the problems makes it difficult to solve. Because of this we focused on this problem: crowd evacuations. We tackled it from several perspectives: from the model, analyzing the dynamics and proposing theories to understand those; from the computing point of view, where the model are thought from their parallelism and the efficiency; and from the DSS thinking on the transfer of knowledge and usage.

When using conventional tools for the simulation of crowds the computational time can augment linearly with the increment of the population. Because of this, there is a need to minimize the time without changing the models. The efficient usage of the resources or the usage of parallel architecture can decrease the time of the computing time. But this is not a free gain, models need to be thought in parallel and programmed in parallel. There has been an effort in crowd dynamics, particularly, in the area of crowds and evacuations it has been more related with computer graphic and realistic visualization of thousands of bots. Thinking about the applicability of the work the goal needs to be the applicability of the knowledge through DSS and workflows and through the analysis of the state of the art and

analyzing the lack in the modern crowd tools.

1.2 Objectives

The present work is focused on the crowd evacuation modeling problem, crowd simulators, and their performance. Crowd models aim to understand the behavior of big populations of people in a closed space. The main goal of this work is to present models and study their implementation on a HPC simulator. The proposed models and methodologies assume that the agents are simplified versions of a person that moves through the space. Fig. 1.2.1 depicts the methodology that proposed in this thesis. The image describes the flow of the simulation process we proposed in our methodology, starting with the input and the configuration until we have the simulation results. This thesis worked on all the parts of the process. We will briefly introduce the methodology.

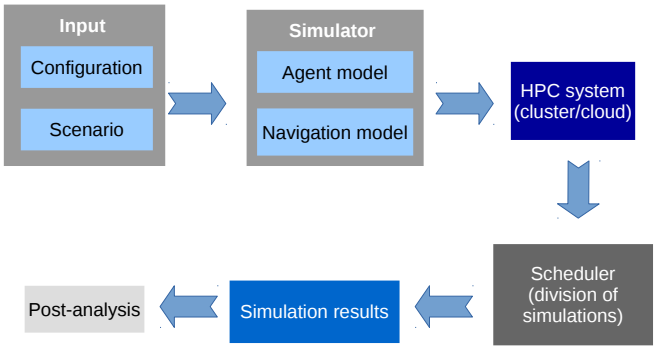

Figure 1.2.1: Evacuation simulation methodology.

In this thesis we will describe two different crowd models that have been implemented inside the simulator. One has been focused on time efficiency and the second has included physical aspects, sociological aspects and psychological aspects that modify the force that the population involved in a high density crowd situation. We will select an HPC system to run our simulation. HPC will allow

us to decrease the total computing time by parallelizing the workload of the simulation. It can be a simulator which runs our model and we can also have another level of abstraction and select the cloud to run the simulation and provide a simple interface for the experts.

We claim that one simulation is not enough to represent the wide range of results. The scheduler divides independent simulation which may have the same or different configuration than the other simulation. The simulation will not have dependency between them and it will produce independent simulation results. Finally, the last step of the process refers to post analysis of the results which is performed by the experts, analyzing the results of the simulation and evaluating. This methodology will be explained with more detail in this thesis and every part of the process will be described.

1.3 Contribution

From the previous introduced issues these are the main contributions proposed by this work focusing on the parallelism and the realism of the model.

- A discrete time model has been proposed, able to handle thousands of agents. For the model was analyzed also a model to evaluate a statistical significant amount of simulations.
- A workflow for crowd evacuations and the execution of cluster in cloud has been proposed. This workflow allows to provide a DSS tool as SaaS running on cloud MPI clusters.
- A parallelization model has been proposed. This continuous model allows to run agents in a decentralized way and move then in a stable free way through the space.
- A model has been proposed to reproduce internal phenomena of crowd events; in particular, crowd turbulence. The model is based on the analysis of high pressure crowds with a high interaction level. It is a parallelizable

model suitable for many core architecture allowing to increase the number of agents avoiding a big performance penalty.

Besides the previously mentioned contributions the tool can be used to several purposes. We list some of them.

- Calculate evacuation time of a scenario for the discrete model.
- Evaluate easily several combinations of configurations of the evacuation.
- Use the decentralized model as basis for new parallel models.
- Test crowds under high pressure conditions.
- Setup a cloud DSS for crowd evacuations.

1.4 Thesis organization

The work presented in this thesis is divided into the following chapters.

Chapter 2, *Modeling crowds and evacuations*: shows the existing models in the state of the art and explains them. Some basic concepts needed to understand the present thesis are explained. The difference and the similarities with the present work are described.

Chapter 3, *Simulation*: Introduces simulation theory, high performance computing architectures and programming models and the application of parallelism to simulation.

Chapter 4, *Crowd modeling*: Introduces the models developed through this work, shows the algorithmic description and describes the parallelism problems.

Chapter 5, *Crowd simulation*: The implementation of the model is introduced, as well as the parallelism issues, DSS workflow and the model to execute a statistical significant amount of simulations. The validation experimentation is also presented.

Chapter 6, *Experimental evaluation*: Evaluated the computational performance of the simulator in several scenarios and configurations.

Chapter 7, *Conclusions and future work*: Presents general conclusions for this work, it makes a summary of the main contributions and opens new line for this work.

2

Modeling crowds and evacuations

This chapter introduces important concepts to understand evacuation and crowd models as well as the ways of representing them. Relevant models for the modeling of these are presented and described as well as methods of representing them, and also specific situations of crowds, crowd disasters and crowd turbulence are explained.

2.1 Introduction

In the last years there has been a growing effort in solving the evacuation safety problem. The research areas interested on the problem vary from engineering, computer science, psychology, architecture or sociology, and each one has been tackling different aspects of it. Inside the evacuation problem lay several subproblems such as: the person model, the psychological components, free navigation through the space, path planning to reach the exit, simulation techniques, performance issues of the simulator, etc. The aim of all these areas and components have common goals: provide a better understanding of the model and have a improved way to implement them with simulators. For this common purpose most of the research considers the same starting premise: evacuate a certain number of agents inside a specific space.

In this chapter several models and approaches to the evacuations and crowd modeling problem will be introduced, varying from complex systems, computer graphics or mathematical modeling. We will first create distinctions among several approaches to modeling crowds and evacuations and later we will extend the considerations also to: parallel problems that need to be solved as path planning, collision avoidance or numerical methods.

2.2 Classification of evacuation and dynamics models

In modeling the problem of representing the reality can be tackled from different perspectives. Each view will have its own pros and cons based in terms such as: complexity, performance, simplifications, aim, etc. First, we focus on the different general classifications of the models and their characteristics.

2.2.1 Macroscopic and microscopic

Macroscopic models use data of the scenario such as flow rate, widths, population, etc., and calculate the results of the scenario globally using a group of mathematical formulas. The navigation of the agents is considered as a continuous flow and

assumes that the behavior of the flow has large scale and that there is no significant change in the decision making process of the individuals which can be simplified. Macroscopic models do not allow to analyze deeply the causes of the phenomena in crowds. To analyze this microscopic models are used, but on the other hand they can be computationally more efficient since the small scale is simplified to a larger scale.

Microscopic models consider the individual interactions of the pedestrians with the environment and the other agents while moving towards the desired goal. They use individual agents with rules, able to interact and make decisions in space and time. These models give a more realistic approach in terms of navigation, movement and decision making in comparison to macroscopic models, but in this increase of realism there is a decrease in performance due to the more detailed configuration and therefore becomes a complex system.

2.2.2 Continuous and discrete

Inside microscopic evacuations models we can differentiate between discrete and continuous. Discrete system are those which state variables change in countable time instants. Continuous systems are those in which the state is changing continuously over time. One of the main approaches in discrete time is cellular automata (CA), which were first proposed by Von Neumann. CA[77][70] are models that evolve in discrete time phases and are characterized by a set of states. The space is divided typically in a grid space and each automata interacts with its neighbors in the grid. The transitions between time steps are done by all the automatas in a step, and will take into account the local state and the neighbour automatas surrounding the target automata. In CA we will consider homogeneous and heterogeneous models depending on the definition of agents. This model has been widely used in evacuation simulation.

Continuous models describe systems with continuum variables. While discrete model can only represent a limited number of positions in the space, continuous model can describe the positions of its components in any coordinate. Therefore

the simulation time and the time step can be modified while in discrete are fixed because the discrete evolution of the simulation. Continuous models gain realism in front of discrete models but they add overhead in the computation.

2.2.3 Behavioral

In some application there is no need to reproduce evacuations that are intended to be validated. Instead of this, the aim of the model is to reproduce behavior of the crowd that will be perceived as realistic. This kind of models are behavioral models that try to reproduce in a realistic way the behavior of the agents which are named "boids" in this case. It has been used widely in animation graphics to reproduce the behavior of boids. One example is the flocking model from Reynolds[60] which models the behavior of birds in a realistic way. Even that these models were not initially intended to act as tools for the evacuation community, they have been widely contributing to the area.

2.3 Social Force

Social Force[29] is, without question, the most influential model on the evacuations research area. It was proposed by Helbing and is a microscopic continuous model. It proposed an explanation for the dynamics of pedestrian under the influence of other people and walls. The model was based on the study of videos and proposed a list of characteristics that emerge from evacuation situations. The list of characteristics described by Helbing is the following:

1. People move or try to move faster than normal.
2. Individuals start pushing, and interactions among people become physical in nature.
3. Moving, and particularly passing a bottleneck becomes uncoordinated.
4. At exits, arching and clogging are observed.

5. Jams build up.

6. Physical interactions cause dangerous pressures.

7. People show a tendency towards a mass behavior.

8. Alternative exits are often overlooked.

This list has been important because it summarized the aspects that are expected for evacuation models involving crowds, physical forces and psychological forces affecting the evacuated pedestrians. After these observations, the Social Force model was proposed. This model describes a set of forces acting on an agent with mass m who wants to move at a desired speed v_i° to the exit e_i°. The interaction forces $\mathbf{f}_{ij}$ and $\mathbf{f}_{iW}$ define how the other agents and walls affect the movement through the space and the trajectory.

$$m_i \frac{d\mathbf{v}_i}{dt} = m_i \frac{v_i^\circ(t)\mathbf{e}_i^\circ(t) - \mathbf{v}_i(t)}{\tau_i} + \sum_{j(\neq i)} \mathbf{f}_{ij} + \sum_W \mathbf{f}_{iW}$$

Force $\mathbf{f}_{ij}$ defines the interactions of agents between them. In the definition of the formula can be seen two components added. The first one declares the interaction with other agents taking into account the distance and several constants (A, B and k) contributing to the body force component. The second part does not allow the tangential movement which will be zero if agents do not touch.

$$\mathbf{f}_{ij} = \left\{ A_i exp\left[\frac{r_{ij} - d_{ij}}{B_i} \right] kg(r_{ij} - d_{ij}) \right\} \mathbf{n}_{ij} + \kappa g(r_{ij} - d_{ij}) \Delta v_{ij}^t \mathbf{t}_{ij}$$

The interaction with wall is described by the force $\mathbf{f}_{iW}$. It uses a similar notation to the body interacting force, but with walls as components, keeping also an exponential repulsion using the distance to the wall to avoid overlapping.

$$\mathbf{f}_{iW} = \left\{ A_i exp\left[\frac{r_{iW} - d_{iW}}{B_i} \right] kg(r_i - d_{iW}) \right\} \mathbf{n}_{iW} + \kappa g(r_i - d_{iW})(\mathbf{v}_i \cdot \mathbf{t}_{iW}) \mathbf{t}_{iW}$$

This model explained self-organized phenomena such as group formation or bottlenecks, and the previous listed characteristics in a clean way. Moreover, is a widely extended and validated model used by a considerable part of the evacuation research community.

After the Social Force was published, there were new contributions based on this work. The Social Force model was parallelized by Quinn[58] for real time simulation of 10.000 agents using space partitioning distributing the space with MPI on a parallel architecture. Apart, the Social Force model has been constantly reviewed by several researchers. One of the most extended revision of the Social Force model is Pelechano's work on HiDAC and MACES[57][56]. Her work is based on virtual crowds and is oriented to computer graphics and it also contributes to evacuation modeling. The model is related to obstacle avoidance, path planning and high densities. She proposed a rich model with a realistic behavior for 3D simulations.

2.4 Crowds

Crowd analysis has been important in buildings construction, public transportation, airports, shopping malls, theaters, etc[10]. There have been disasters along the history, increasing the analysis of the crowds during the last century. Fig. 2.4.1 shows some of these disasters that have happened in different scenarios. The images give an idea of what crowds mean in terms of size and complexity and how it affect to different parts of the world. Crowd evacuations is a special case inside pedestrian evacuations. Besides the amount of agents involved in the simulation there are other important factors that impact on the model: crowd special phenomena. When there are hundreds or thousands of persons in a dense mass the events that happen there are different from a low density evacuation. These phenomena con vary from stop and go waves, laminar behavior, crowd turbulence, stampedes, etc.

Figure 2.4.1: Crowd events and places where people resulted death and injured: (a) Mahamaham Festival, Kumbakonam [69]. (b) Hillsborough disaster in the newspapers[53] . (c) Love Parade in 2010[1]. (d) The way to Jamarat bridge [78].

2.4.1 Crowd disasters

Crowd disasters are a historical problem in crowd situations. They have been caused by stampedes, high dense situations, bad planning, etc., and they injured people and some cases were the cause of death. We list in table 2.4.1 historical crowd disasters[13][30]. They are divided in historical disasters and disasters in the last 10 years. Those show how in the last century the crowd disasters did not stop occurring with hundreds and in the worst cases thousands of deaths. In real complex and crowded places such as Mina in Saudi Arabia where millions of pilgrims travel, there have been several disasters over the years due to the complexity of the situation and the big amount of people who attends. The second part of the table demonstrates how in the 21st century disasters did not stop occurring. The number of deaths in the list varies from a units to thousands. Also, normally

in these situations there is a big number of injured people. The table shows how these cases happen all around the world and are not restricted to a unique pattern or region.

2.4.2 Continuous crowds

The term "continuous crowds" is usually referred to the continuous movement of pedestrians. There have been several models representing the navigation of big populations of boids through the space. Treuille[72] presented a real time crowd model with collision avoidance under dynamic conditions based on limited knowledge of the boids and dynamic paths based on potential fields driving to an efficient algorithm for continuous crowds. Although pedestrian models have been focused on the optimal real time simulation of boids, these have found representation of real phenomena such as line formation and have based part of their work in contributions of evacuation models. These models are focused on the graphics and rendering process, where thousands of agents are moving in an intelligent way, avoiding obstacles and agents optimized to reduce the computational complexity. They are able to handle thousands of boids moving in a space. This technology is also used in video games to simulate population in cities, an army, etc.

2.4.3 Crowd turbulence

Crowd turbulence is a special phenomenon that had turned events into disasters, such as Love Parade (2010) or Hajj (2006). Helbing wrote an empirical study of the Hajj disaster[31] where he analyzed the videos of the Jamarat bridge were 345 people died a day more than 3 million people assisted. He extracted conclusions from the video analysis such as how the turbulence happened and the he proposed the formula of "pressure" to define the behavior of the mass. Posterior research proposed a model to reproduce crowd turbulence and using the "pressure" formula to validate it. The "pressure" formula will be described in section 5.4.3.1 and used to explain crowd turbulence of our model. The Social Force was extended by

Table 2.4.1: Major crowd catastrophes.

Year	Place	Venue	Deaths	Comment
1863	Santiago, Chile	Church	2000	Emergency egress
1881	Vienna, Austria	Theatre	570	Emergency egress
1903	Chicago, USA	Theatre	602	Emergency egress
1943	London, UK	Tube station	173	Ingress
1961	Rio de Janeiro, Brazil	Circus	250	Emergency egress
1964	Lima, Peru	Stadium	318	Riot
1968	Buenos Aires, Argentina	Stadium	73	Egress
1971	Glasgow, UK	Stadium	66	Egress
1974	Cairo, Egypt	Stadium	48	During event
1982	Moscow, USSR	Stadium	340	Egress
1989	Sheffield, UK	Stadium	96	During event
1989	Mina, Saudi Arabia	Pedestrian tunnel	1425	Ingress/Egress
1992	Kumbakonam, India	Outdoor Festival	50	During ceremony
1994	Mina, Saudi Arabia	Holy site	250	During ceremony
2006	Mina, Saudi Arabia	Jamarat Bridge	345	During event
2006	Pasig City, Philippines	Stadium	78	Ingress
2007	Chongqing, China	Supermarket	3	Offer
2007	Sunchon, North Korea	Stadium	6	Egress
2008	Mexico City, Mexico	Nightclub	12	Riot
2008	Himachal Pradesh, India	Temple	146	Egress
2008	Jodhpur, India	Temple	224	Bomb rumor
2009	Abidjan, Côte d'Ivoire	Stadium	19	Egress
2009	Xiangxiang, China	School	8	Egress
2010	Duisburg, Germany	Tunnel	21	Ingress/Egress
2010	Kunda, India	Temple	71	Egress
2011	Haridwar, India	Holy site	16	Ingress
2011	Kerala, India	Holy site	102	Egress
2012	Cairo, Egypt	Square	3	During event
2012	Madrid, Spain	Stadium	5	Egress
2013	Abidjan, Ivory Coast	Stadium	60	Egress
2013	Santa Maria, Brazil	Nightclub	242	Egress
2013	Uttar Pradesh, India	Railway station	36	Egress
2014	Shanghai, China	Outdoor event	36	During event
2015	Cairo, Egipt	Stadium	28	During event
2015	Mina, Saudi Arabia	Holy site	2262	During ceremony

Yu and Johansson[82] to model turbulent flows. It extends the repulsive force of the agents. They added extra terms to the repulsive force improving the previous Social Force model. This new approach considers the following formulas to model the repulsion creating a more complex way to solve the overlapping between agents and the "Social Force". The new constants used are based on the distances between agents and the interaction angles.

$$\vec{f_{ij}} = F\Theta(\varphi_{ij})exp[-d_{ij}/D_0 + (D_1/d_{ij})^k]\vec{e_{ij}}$$

$$\Theta(\varphi) = (\lambda + (1-\lambda)\frac{1+cos(\varphi)}{2})$$

Further studies such as the analysis of Love Parade by Helbing [28] also detected crowd turbulence producing high pressures that crushed the chest of several assistants. Another model was proposed by Golas[25] which is based on Fluid Implicit Particle, a PIC code. The goal is to represent the crowd as a continuum with collision avoidance and computes friction stress.

2.5 Navigation and collision avoidance

While pedestrians walk through the street they need to move in different directions, cross other pedestrians, stop if there others are stopped, avoid bins, posts, and other obstacles. Therefore agents need to have the intelligence to avoid other agents and obstacles while they are following their path. To solve these problems we use navigation and collision avoidance algorithms, especially in pedestrian dynamics. These algorithms are specially used in computer graphics and robotics. One of the more important algorithm in this area is Velocity Obstacles (VO)[21]. VO is a behavioral model based on the prevision of the future position of the agent and correcting the trajectory when two or more agents share this trajectory. It has been one important navigation algorithm used[26], reviewed and extended by algorithms as Reciprocal Velocity Obstacles[75] or Hybrid Reciprocal Velocity Obstacles[65]. VO is expressed using Minkowski notation to denote the inter-

sections of the areas of the navigation boids. The main idea is: each agent has a cone defining the possible future positions; the agents try to move to areas where their future positions do not conflict with others, moving always to free areas or stopping if they cannot.

$$VO_B^A(v_B) = \{v_A | \lambda(p_A, v_A - v_B) \cap B \oplus -A \neq \emptyset\}$$

In more detail we can consider two agents named A and B. We use Minkowski notation to express the cones defining the possible areas where $B \oplus -A \neq \emptyset$ denotes that there is no intersection between both areas. The algorithm checks if there will be a collision in future time steps and then correct the navigation for the next step providing free collision navigation.

2.6 Modeling human psychology

Even though some macroscopic approaches that model the crowd have minimized the impact of sociological components and individual attributes of the occupants, these can have a direct impact on the final results of an egress situation. We describe some psychological and sociological aspects in risk situations and how people had behaved in those.

2.6.1 Sociological components

Mass panic, selfish behavior and self-preservation has been considered the normal response to potentially dangerous situations. Current literature has demystified this belief, showing how aid and non-selfish behavior is more dominant[50]. Also social contexts such as being close to family or people with any emotional attachment calms down the person and leads to a gregarious behavior in a potentially stressful situation[50]. Moreover, distance from this emotionally attach people is a greater stress factor than any physical danger.

The reaction from different countries and societies differs and can have a direct impact on the results of a potential risk situation. An example to illustrate the re-

action of different societies to panic situations are the "War of the Worlds" Orson Welles broadcast and a Swedish broadcast of a radioactive leak. In October 1938 Orson Welles described on the radio how Martians were invading the Earth during prime time reporting it in the news. Between 6-12 million people heard the program and 28% believed it[7]. And of these, the 70% were "excited" or afraid[46]. On the other hand, a radio broadcast informing of a radioactive leak in Sweden from a nuclear plant, had different results. Of the interviewed, only 1% of them confessed to react behaviorally to it[62]. No one felt panic as opposite to what newspapers said. This is a sample of how under similar conditions react differently to its sociological components. Therefore this behavior needs an analysis of the population and the society. The evacuation will be different in different countries and in different ages. Also, population will behave differently in a known space such a residential evacuation than in a stadium evacuation.

2.6.2 Psychology and panic

Even in events that are considered by media as panic and stampedes, such as the Who concert stampede (11 deaths), people tried to help each other and posterior research concluded that these situations were not stampedes[39]. To clarify what is panic we will cite its definition as described by Mawson[50]: *The term "panic" refers to inappropriate (or excessive) fear and/or flight and highly intense fear and/or flight. instances of panic are difficult to identify in practice. The judgment of panic is usually made retrospectively, especially if serious loss of life occurred*. Therefore, we assume that panic situations will make people to put themselves as priority over the other and act in ways that may harm others.

Current research rejects the term of mass panic[2][37] which is historically related to uncontrolled emotion, and selfish behavior. These are now considered as irrationals theories of the crowd which assume the loss of individual personality. In Hillsborough disaster with 96 deaths, testimonies related that there was co-operation and order behavior even when the threat of death was present[8]. Thus, humans do not always act selfishly[76].

The fact that these studies demonstrate that panic and irrational behavior is seldom makes changes the approach of some evacuation models as they are considered as "panic situations". It would be an easier explanation for selfish behavior, panic and stampedes to disasters but the posterior analysis of these situations show that it is rare. There are factors such as social identity which mitigates the effect f crowd risk such as social identification [3]. Psychological membership to a community can have benefits while other people identify themselves with the crowd reducing the risk acts and moderating their behavior. Pro-social behavior in evacuation is often found[16][15]

2.7 Mass forces

In the Hillsbrough disaster, the barriers collapsed due the force of the crowd. Posterior analysis showed that the barrier was able to hold up to 6kN/m [27][54]. This force was the result of all the force acting in one direction. This range of forces can be dangerous and drive to a potentially pernicious situation. These forces have been studied and particularly Henein[33] proposed four general principles to describe and model them:

1. Force is directed. It is applied by one agent to another in a particular direction. Forces in models must be implemented as vectors rather than scalars.

2. Force carries consequences. Force is not an invisible field that guides cognitive decision making; rather, once exerted it has measurable effects that are outside an agent's control.

3. Force is propagated. Force in a crowd model must be transmitted through the crowd in the direction of the force, and that it is additive (vector-wise) when encountering other forces

4. Force is generated purposefully. Force is not simply an emergent property of masses of people. People exert force in some circumstances and not in others.

2.7.1 High density

High density crowds are a special kind of crowd where pressure may be higher and the consequences can be different than in other crowds. There is more pressure in the agents and the risk of crush is high. In the analysis of crowd turbulence by Helbing, greater than $10 persons/m^2$ persons were observed when the normal condition should be 4 in average[31]. It also analyzed that when local density of $6 persons/m^2$ was exceeded, the local flow decreased by a factor of 3 or more. This pressures are dangerous and the density may propagate the forces of the bodies and harm the assistants. For example, in the case of the Madrid Arena disaster, survivors described how they were not able to walk. They were dragged by the force of the mass and some bodies felt and crushed bodies that were at the bottom of the pile. In these cases the will and forces of the agents can be not significant. But the addition of all the forces becomes critical. An analysis and modelling of pressure[23] showed how the pressure affected the survival of the people. Ibrox Park soccer stadium incident produced asphyxia, with a 3 meters pile of bodies supporting 3600-4000 N. The addition of forces causes enough pressure to compress the chest and produce asphyxia or in worst cases, "chest crush". These forces are dynamics since all the affected persons try to gain space to breath. Analysis from videos such as the love parade disaster[9] show that people tend to gain free space as it was previously concluded in the first Social Force paper presented by Helbing[29].

2.8 Modeling with DE

A model emulates the behavior of the real world, and the behavior of reality is most of the times expressed or it can be expressed in differential equations (DE) or it can be expressed. For this reason they are important tools used in scientific simulation. In science and engineering mathematical models and physical models are often expressed in form of equation with derivatives on its declaration.

Partial Differential Equations (or PDE) are differential equations with more than one independent variable and its derivatives. PDEs are widely used in sci-

entific simulation and the description of processes particularly in physical process to define a model set in space how evolves in time. In the case of evacuation there have been approaches to the usage of PDE to solve the movement of the population. It has been applied in population dynamics[40] with gas-kinetic or fluid dynamics approaches. There is an important branch of crowd dynamics theory which creates an analogy between the flow of the crowd and the Navier-Stokes equations[32]. Navier-Stokes are a set of PDEs used to model computational fluid dynamics. These equations have high computational demands but in the case of crowds they have been used for 2D problems. Hughes made contributions on the modeling of the flow of pedestrians as PDE[35][34] focusing on the dynamics of the mass as a global behavior using a macroscopic approach. Moreover, Social Force used the differential form of Newton's second law of motion as starting point for his model.

Ordinary differential equations (or ODE) are differential equations with one or more functions with one independent variable and its derivatives. These equations are easier to solve since the number of independent variables used is lower than in PDE. Even though the usage is lower in scientific computing they are also widely used. ODE will be used in the present work to define the continuous crowd model.

There are several numerical methods to represent ODE and PDE. In the finite differences method, it is based on a differential formulation of the DE. The space is divided in neighbor points and it is used an expansion of the Taylor series between two points. Other methods are the finite element method and the finite volume method which allow to solve PDE for unstructured grids.

2.9 Conclusions

In this chapter we introduced a range of approaches to modeling crowd evacuations as well as some problems related to it. We saw that there is not a unique way to represent them (discrete, continuous, microscopic, and macroscopic) and choosing between one and another is a trade-off. Crowd are complex situations and have led to several disasters through the history and there are some events

that only occur in this situations. One of the most important models in evacuations is the Social Force model which is based on the forces that agents and walls interact on every agent. The psychology of the crowd can modify the results of an evacuation process and panic should not be considered as the obvious reason to crowd disasters.

3

Simulation

This chapter, introduces concepts on computer simulation as well as their approaches on the computational models. Also high performance architecture and computing models that allow to implement time demanding simulations are discussed. HPC simulators and techniques to reduce the computing time of those are introduced.

3.1 Introduction

Most real world problems cannot be solved analytically. To solve them we use machines that allow to achieve an approximated solution. Computer simulation is the technique used to emulate the behavior of a system using the power of computers. To perform this action we need a computer program which will be called simulator, and a computer or set of computers where the simulator will be executed. Moreover, one computer cannot be enough to solve some kind of problem which may demand more computing time. In these cases we need to parallelize the computing process in order to reduce the total execution time. For this we will use parallel simulators.

3.2 Computer simulation

Simulation modeling is the process of creating a computerized model from the description of a theoretical model. Therefore is the process of translating the model to the interface that provides a computer: a discrete machine that performs arithmetic-logical operations, input/output interactions and conditional and unconditional jumps. Simulation has been growing the last years with scientific research. At the beginning of its history, computer simulation was strongly related to military research. For instance, in the Manhattan project (1942-46) a simulator was built in order to simulate a model of a nuclear detonation and understand what could happen[59]. Simulation permits to understand complex interaction within the system and allows to go forward and back in time, analyzing the behavior as well as trying new concepts and answering question of the type of "what if...?". Allows to simulate high complex system that otherwise we will not be able to understand. We use it to predict the future, to understand the pass or even to emulate an existing system. Simulators are also used for training users in complex systems as in the case of flight simulators, where the simulation cost is very small in comparison of interacting with the real system. Nevertheless the simulator will be as good as the model and the configuration. Simulation will not solve all kind

of complex problems and will not provide easy answers. They need to implement realistic models, be properly coded and also have accurate configuration and data.

3.2.1 Simulation classification

There is not a unique way to implement a simulator. Its scheme will depend on several factors as the problem, approaches to the problem, design decisions, performance, etc. Simulation can be classified depending on several attributes. Fig. 3.2.1 depicts the taxonomy of the categorization based on determinist/stochastic, static/dynamic, continuous/discrete properties.

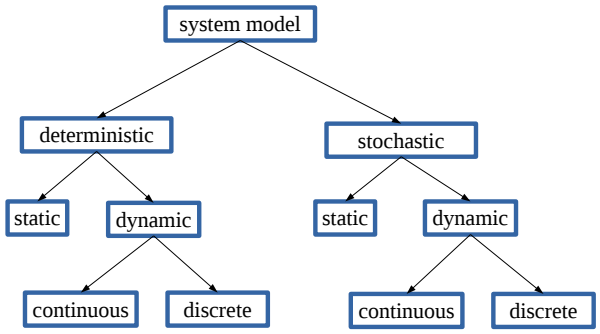

Figure 3.2.1: Simulation model taxonomy.

Deterministic and stochastic

A stochastic system has random patterns determined by probabilistic components of the model. Small variations in the system can cause big variations over time. For example stock markets are modeled as stochastic with random patterns modifying its behavior, the movement of cars in traffic, or the probabilities that a cancer cell will reproduce. Therefore even knowing the range of possibilities, there is an unpredictable output of the simulation because it will vary. In this we can find Monte Carlo simulation which is based on an iterative simulation with randomness on the

behavior. Deterministic systems do not have arbitrary patterns and will not have random behavior. Due this, the output of the system will always be the same for a given input. Therefore it will be more predictable than a stochastic system.

Static and dynamic

A static system does no evolve in function of time. In some cases we could consider that is a simulation which is done in steady state, only for one simulation step or in other systems, time is even not a parameter. On the other hand, a dynamic system evolves in function of time. Hence, the state of the simulation will vary and evolve between simulation time steps.

Continuous and discrete

A simulation is continuous if its definition is expressed in function of continuous time and the state is constantly changing. On the other hand discrete simulation are those that advance in discrete periods of time and the change is only produced on these discrete steps.

3.2.2 Scientific simulation categories

Here we describe some approaches to modeling physical system using simulation. The categories vary on their approach to model the problem. The described categories are considered important in the area of evacuations and simulation.

Continuous dynamic simulation

This is one of the biggest areas inside scientific simulation. Its description is based typically on differential equations and algebraic equations. As described previously a large amount of physical systems can be modeled as a set of differential equations that evolve in time and they are implemented using methods such as finite volumes, finite elements or finite differences. The equations will define state variables, boundary conditions, how the system changes between time steps, etc.

Even that continuous simulation represents continuous system, the solution will always be approximate. The complexity of continuous simulation is normally high with high computer performance needs. In the case of fluids, which have been compared to the movement of the crowd[32], we have PDE that define the macroscopic behavior of the crowd and all the set of equations need to be solved for every time step and for every point in a determined space defined by a set of points.

Discrete event simulation (DES)

Discrete event simulation (DES) describes a model using a discrete sequence of events that determine the evolution of the system. Transition between states are triggered by events and no update of the system is done in between. The system has a set of variables which are updated at every event. Due to their discrete nature are easily adaptable to computing systems and there is a considerable amount of formalism on this area. DES is a very important and extended category in scientific simulation. It allows to model finite state machines, Petri diagram, queues, networks, etc. For example, in the case of Petri diagrams we will tokens, transitions, places and arcs. The token, which are the representation of item move between places according to the event list. Therefore, there will not be a simulation time that determines the evolution of the system, but they will have events that will set transitions between simulation states.

ABM, IoM

Agent based modeling (ABM) is an approach to modeling systems composed of individual, autonomous, interacting agents which are declared by behavioral rules and attributes. ABM is based on the representation of global behavior from the rules given to individuals and it enables us to see the consequences on a macro level of the interactions on a micro level. In ABM, behavioral rules are the set of rules that define how the agent act at every time step. The behavior will depend on the environment and the attributes. Attributes are state variables of every agent and independent from others which are updated by behavioral rules. In the par-

ticular case of evacuations these attributes may be, *size*, *age*, *exit*, v_{max}, etc. These attributes make different and particular every agent by having different attributes. The environment of the person are the elements which interact with the agent. For example in the case of evacuations the environment can be defined by walls, obstacles, signals, etc. There will be a set of individuals with their behavior and attributes and the environment. We can have a vision of how the individuals interact (homogeneous or heterogeneous agents) with the environment and the other individuals, in discrete times. Every agent is ruled by the same laws. ABM allows us to analyze how the system evolves from the rules of the individuals and their interaction with others and the environment. From the beginning, the output of the system is unknown, even with a model and a known configuration. We cannot know the output of the system analytically, and because of that we will need to simulate it. In the case of crowd evacuations it has been used previously. This approach can be considered more realistic than a system of equations solved. This approach is then closer to the reality. ABM as good aspect, offers more flexibility and the availability of setting different behaviors depending on the environment and allowing heterogeneous agents and environments. Agent's attributes do not depend on others and agent's state variable are updated every time step. It allows to analyze and evaluate the internal evolution of the system and have more information from the state update of the agents. In the case of ecology, these models are known as IoM (Individual oriented Models). Are applied in herding, flocking or other natural system.

3.2.3 Simulation time

Simulation time is a synthetic time used by the computer to have an internal way to approach a measure of the time of the real system. It is also different from wall time which is the total execution time of the simulator. Simulation time advances on time steps, which are the discrete way in how the simulation time advances. Choosing the adequate time step impacts directly on the total execution time of the simulation. A small time step will give more information but the simulation

will take longer. A big time step can make the simulation run faster but may lose information. For problems where there is a wide variance in a short period of time, such as combustion simulation, a short time step is needed. In cases such as N-body collision if the collision time can be calculated there can be used dynamic simulation time, optimizing the total execution time.

3.3 High Performance Computing

Some problems cannot be solved in computationally conventional ways because the total execution time will be too long. Therefore there is a need to reduce the total executing time and make it faster. The aim is to conserve the problem and speedup the execution. For this we have High Performance Computing (HPC) which maximizes the usage of modern hardware and software using parallelism. Parallelism consists on dividing a part o the problem in pieces of work in an ideal way that dividing it into N parts will take $total\_time/N$.

3.3.1 Parallel architectures

Parallel architectures can be divided by their properties. A well known taxonomy used to classify the computing architecture is the Flynn taxonomy which uses two dimensions (instructions and data) to classify four architectures types.

- *SISD (Single Instruction Single Data)*. Is the most conventional architecture where there is one flux of data computed with one flux of instructions
- *SIMD (Single instruction Multiple Data)*. In the case of GPUs the term SIMT (Single Instruction Multiple Thread) is used because all threads are executing the same time with a shared program counter with different data. A classical example of this are vector processors.
- *MISD (Multiple Instruction Single Data)*. The same data is computed differently from the processing units. This architectural is only theoretical and has no commercial usage event that there have been built prototypes.

- *MIMD (Multiple Instruction Multiple Data).* There is a parallelism of the process and the data.

3.3.1.1 Shared memory

Shared memory architectures consist on a set of processors that communicate between them using a common memory. Fig. 3.3.1 shows how independent cores with a cache share a memory that is accessed by a memory bus. Inside shared memory architectures we can divide them depending on the access to memory: NUMA (Non Uniform Memory Access): the time to access the memory is not constant and the latency will vary; UMA (Uniform Memory Access): constant latency to access memory.

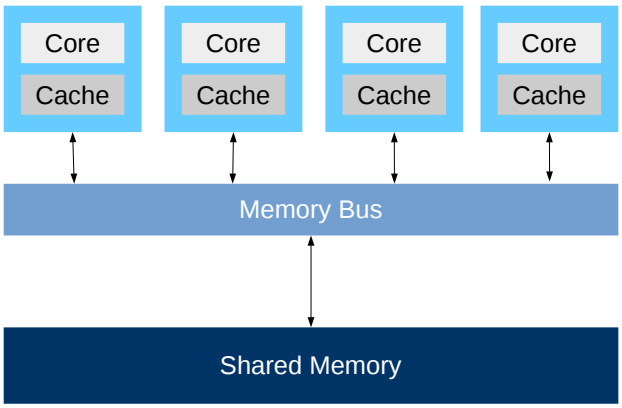

Figure 3.3.1: Shared memory architecture overview.

An example would be a multicore architecture where independent processors with different processing contexts share memory addressing and can modify common variables by having a shared space using threads instead of process. Other architectures such as GPUs follow a SIMD architecture where there is a common context inside a same set of cores known as SM (Streaming Multiprocessor) which share program counter, and all the cores have a common memory space which can

be accessed by all the threads. Coprocessors such as XEON Phi are a set of low-energy processors with Intel ISA that solve vectorized problems.

3.3.1.2 Distributed memory

Distributed memory architectures are based on independent processors with private memory which use an interconnection network to communicate. Typically the units of distributed memory are nodes. Every node has processors, which may have shared memory as well. Fig. 3.3.2 shows a system of nodes with one core which own an input/output device to access the interconnection network to communicate with others.

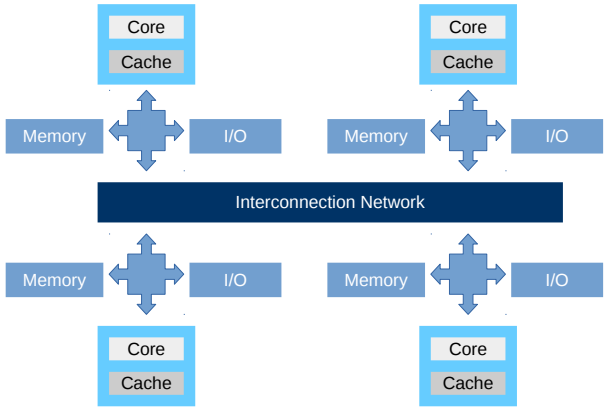

Figure 3.3.2: Distributed memory architecture overview.

3.3.2 Parallel programming models

Programming languages allow to give instructions to parallel hardware on how to coordinate all the computing devices in order to solve a common problem. Parallel programming models have also a difficulty added which is providing an interface with complex hardware in a simple way but without losing performance on the compute.

3.3.2.1 Shared memory

OpenMP is an application programming interface (API) for multi-thread programming. It is used in multicore architectures, and also, more recently, in Xeon Phi. It is called using the format of precompiler directives, so if OpenMP is not processed, the API directives will be ignored. The execution model of OpenMP is fork-join where the task is divided into n children threads to finish collecting the solution. Syncing and passing values is done through sharing a set of addresses in memory. POSIX threads (pthreds) also provide a library for thread programming to create and control their flow.

3.3.2.2 Coprocessors and accelerators

CUDA is a parallel computing platform and API created by Nvidia. It allows to program a parallel kernel which runs on Nvidia GPUs. CUDA parallel regions are named kernels, which are functions callable from the host and run on the GPU. The kernel will use threads, which is the basic unit of work of the GPU. All threads will run at the same time in the GPU and this usage is called thread-level parallelism. CUDA is widely used in supercomputers and it provides an easier programming interface compared to other APIs such as OpenCL which makes a faster development, but, on the other hand, it will only by supported by Nvidia hardware.

OpenCL (Open Computing Language) is an open standard for general parallel processing aimed to support different platforms, ranging from multicore, GPU or even FPGAs. As opposite to CUDA which is intended for Nvidia platforms, the same code in OpenCL can be ported to different platforms. The standard defines an API which is implemented by the hardware developers providing a homogeneous view of the resources. When programming in OpenCL there is a code for the host (CPU) and a code for the device in the format an OpenCL kernel, which is loaded in execution time. The host code allocates the memory, configures the OpenCL environment and calls the kernel which runs in parallel on the device.

3.3.2.3 Distributed memory

Programming model based on distributed memory architectures specify how nodes process communicate with others using messages. Variables are only shared through the network using messages. The programmer needs to build the architecture of the execution, the master and slaves, synchronization, etc. Event that there are other specifications for distributed memory as MPI (Message Passing interface) became a *de facto* standard in the community. MPI is a portable and flexible specification based on the efficient use of architectures and high performance systems. There are libraries that implement these specifications such as OpenMPI, MPICH, Intel MPI, etc. MPI has been used in most supercomputers of the Top500 list[71] from middle 90's, and it is still one of the most extended in HPC.

3.3.3 Hybrid computing

The present and future of HPC goes toward the hybrid computing, mixing architectures and programming models. In a modern node of a supercomputer of the top500 list we can find several cores, in several CPUs, which inside have parallelism in the data and instruction access, superscalar execution, vector SIMD operations, a GPU or vector processors with Xeon Phi that may be connected via an interconnection network to nodes with different hardware characteristic to their own. Programming models are now mixed in modern HPC applications and the hardware usage is optimized.

3.3.4 Parallelization techinques

Once the region that can be parallelized has been located inside the model, a paralellization technique will be chosen. The technique will depend on the dependencies and properties of the algorithm, design decisions and the architecture. When parallelizing the problem at times there are dependencies and some simplifications are made to extract as much parallelism as possible. At times there are hard dependencies but simplifications are made in order to have a parallel problem. In other

cases such as matrix multiplication the problem is implicitly parallel. There are several standard techniques on how to divide the work in pieces and distribute those.

- *SPMD* (Single Program, Multiple Data) is a technique that consists on running different inputs of the system in different processors to achieve results faster.
- *Master worker* is a communication model where a master divides the work among slaves and once they have finished will communicate it to the master to deliver more work.
- *Fork-join* is a parallel design pattern which divides the work in several independent branches to create independent solutions that are computing in parallel and later merge these solutions into one. Fork-join model follows a divide and conquer pattern.
- *Pipeline* is a technique used to overlap the execution of tasks that depend on each other, but which execution can be overlapped and the output of one will be the input of another. It is commonly used by instruction level parallelism to speed up the processor execution.

3.3.5 Cloud

Cloud allows to provide services on demand by accessing to them through an Internet connection. It allows to have complex architectures and setups paying for the infrastructure providing IaaS (Infrastructure as a Service). Cloud has become a new service in simulation and HPC. An HPC cloud provides dynamically scalable resources which allows to have HPC on demand. Platforms such as StarCluster[51] developed by MIT allow to run HPC simulation on the cloud in a simplified way for the scientists. Several simulation projects have researched on the performance of this new approach to provide simulation services, such as biomedical HPC applications or the Flame ABM framework [24][43]. Even though these advantages are good to consider, it has some disadvantages such as

the irregular access to the file system and non-uniform behavior of the network which may penalize the applications and computing time which will end up increasing the price of the computing.

3.4 HPC simulation

Simulation has been related to HPC since the beginning of computational simulation. Lewis Fry Richardson imagined on 1922 a "forecast factory"[48] with more than 64.000 human using their calculators at the same time to forecast the weather of the planet on real time. That could be considered an early approach to supercomputing simulation and shows how the computing needs have been needed from an early stage. Also, in non-deterministic models we may need a big set of simulations to have a significant amount of results.

3.4.1 HPC in scientific simulation

Almost all applications that run on supercomputers are simulators. The goal of scientific simulation in supercomputers is to predict, analyze processed that otherwise could not be understood, analyze big amounts of data, reproduce complex experiments, etc. To have a better understanding of the applications that are used in supercomputing simulation we list some of them separated in areas.

- *Bioscience*: drug design and applications such as genetic alignment or protein folding.
- *Chemical engineering*: combustion, chemical reactions, etc.
- *Economics*: risk analysis of markets, automated training
- *Geoscience*: oil and gas exploration. For example big oil companies use supercomputer to analyze data of the bottom of the ocean explored previously using waves.
- *Mechanics*: solid state physics, computational fluid dynamics, etc.

- *Weather and climate forecasting*. Climate change analysis, volcano dust.
- *Physics*: plasma physics, cosmology or astrophysics models.

3.4.2 HPC in agent dynamics

The literature on computational crowd dynamics has been improved by the game industry, aiming to find realistic behavior of boid agents moving through the space[26][66]. These approaches allow to parallelize more easily because they break dependencies. Another important field making contributions has been robotics, where independent robots have to move freely through the space avoiding collisions and being independent from any centralized decision in a similar way that evacuation agents[42][81][14]. In the area of evacuations, free navigation and HPC there are several approaches with GPUs. The approaches have been more related to realistic behavior of boid agents moving freely among the space that trying to provide a model for crowd evacuations and also has been addressed as a rendering problem. For instance the Bleiweiss[5] of GPU works on free navigation including models such as path planning. Richmond[61] proposed Flame GPU template where he creates a GPU tool able to produce templates for SIMD problems and visualization. Other approaches in the case of ABM model show the performance of the free navigation of independent agents through the space[18].

3.5 Simulators and commercial software in evacuations

In the case of evacuations, crowds and pedestrians there are several academic and commercial simulators published. They differ on their approach to implement and solve the crowd and evacuations problems. We analyze the models and also make use of a review made by NIST[45] to categorize and describe them.

- *Simulex*[64] allows to simulate crowds in complex 3D structures. It is a partial behavioral model, considering distance between persons to set agents speed. The goal of the simulator is to allow to set up occupants inside a building and simulate the movement during an emergency.

- *STEPS*[68] permits complex 3D spaces. It is an ABM tool, microscopic model, where the individual movement is modeled, aimed for evacuation situation and for normal situations. It allows to create groups use more complex elements such as elevators.

- *Oasys MassMotion*[67] is a 3D tool compatible with 3D software such as 3D Studio Max, able to configure and build complex scenarios and simulate navigation of pedestrians.

- *JuPedSim*[41] is a public simulator developed by Jülich Research Center. It allows 2D scenarios and agent move following Voronoi distance and a defined path. Provides data sets for validation, calibration and it is composed by a set of packages for the analysis of the simulation: JPScore, JPSvis, JPSed and JPSreport.

- *PathFinder*[17] provides egress analysis in complex 3D structures. Agent's positions are tracked by the simulator. The occupants move to the exits following certain constraints. It is compatible with fire simulators.

- *Myriad II*[36] manages 3D complex buildings. Calculates congestion and its severity and gives an overview of movement of the occupants. Its purpose is to assess security, calculate times, number of interactions and determine exit capacity.

- *Exodus*[20] is a behavioral model used for evacuation analysis and pedestrian dynamics. It considers people-people, people-structure and people-fire interactions. The model consists on submodels: occupant, movement, behavior, toxicity, hazard and geometry. I allows to simulate toxic gases, smoke and temperature.

- *Legion Evac*[19] is a behavioral model able to handle CAD structures, and planned to optimize crowd behavior and interaction between individuals to study evacuations. It is aimed as a consultancy tool. It calculates counter

flow, and evacuation times. It is compatible with Fire Dynamics Simulation software to simulate evacuation under fire conditions

3.6 Conclusions

We discussed how computer simulation helps to understand real systems and contributes to take decision and improve the existing knowledge. Simulation is used in a wide range of areas, and in some of them there are specific computational requirements that demand to decrease the total execution time of the simulator to have a feasible total computing time. For this we described how the parallelism is used on HPC architectures using parallel programming models. In the particular case of evacuations and crowds some approaches have been developed for the optimization of the simulation but they have been more related to computer graphics.

4

Crowd modeling

This chapter, presents the two models developed during this thesis: a discrete time model and a continuous time model. The internal behavior and the pros and cons of both are discussed. Moreover we describe the algorithms that defined their behavior as well as the internal aspects.

4.1 Introduction

Models allow us to interact and test the system without using it. During this thesis we developed two crowd models both using ABM approaches. The first model was focused on the representation and simulation of crowds and the collision avoidance of scenarios and agents in an efficient way using a discrete time model. The second model aims to reproduce internal behavior of the crowd and increase the realism using a continuous time model. We lost performance, but, on the other hand we won realism. These models were proposed with the aim of having models able to handle thousands of agents managed in an efficiently.In the following sections both models will be described and detailed.

4.2 Discrete time model

The first model to be developed was the discrete time model which is based on some concepts of CA and implements a crowd model based on a grid space and agents of a fixed size which are moved from cell to cell using intelligence based on their knowledge. These techniques have been also used in previous studies such as hybrid models with fire[70] which named to these agents as Intelligent Agents dividing between the agent sub-model and the environmental sub-model. Our model is computationally efficient and is able to handle tens of thousands of agents which was the purpose of the model. We will divide the explanation of the model between the agent behavior and the path planning and collision avoidance model which will be the navigation sub-model where the intelligence of the agent will remain. Both models together allow the representation of crowd dynamics.

4.2.1 Agent sub-model

The computational model of the agent is described by a set of states and variables. Fig. 4.2.1 describes decision between states and the flow diagram and Tab. 4.2.1 the meaning of the states and variables. The graph illustrates the behavior for one agent in one simulation step. The movement of the agent is described by the will

of approaching the exit and moving to free spaces, which is the condition of the movement. Once the agent has reached the exit, it is released and not considered again by the simulation process, decreasing the total active population of the simulation. In the case that we cannot approach the exit, then the individual will wait for the people that are between them and the exit to move. This logic is applied in every simulation step in the time-driven simulation. All the states are internals of every individual. As seen previously, the rules given to the ABM are what determine the behavior of the group. The goal of the individual is the exit and the minimum distance determines, by default, where the person is going. The simulation process will finish when the state of all the agents has been set to Q_3.

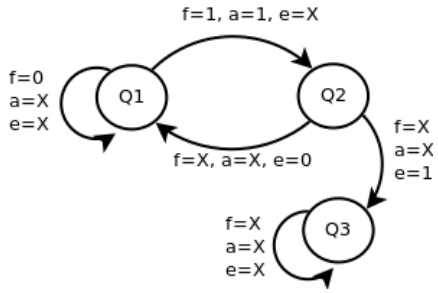

Figure 4.2.1: Model's state diagram.

Table 4.2.1: Discrete model state diagram description.

Type	Attribute	Description
State	Q_1	Stopped
State	Q_2	Moving
State	Q_3	Evacuated
Variable	a	Approaching the exit
Variable	e	exit reached
Variable	f	free patch

In the discrete time model the space is divided in grids. These grids represent a space of 40x40cm assuming that the space used by a person is this. Locally the individuals will move to another path that fills their objective function: approaching

the exit. In this case we will only consider single-story spaces. There are two kinds of neighborhoods 4-connected (Moore Neighborhood) and 8 connected (Von Neumann neighborhood). In our model we use 8 connected cells. The three cells that are in front of the agent will be evaluated and there will be random patterns of diagonal movement spreading agents and not concentrating them in a vertical line. The cells cannot be evaluated in a certain order, otherwise all the population will move to one side.

The model presents phenomena such as bottlenecks, clogging or arching close to the exits. Mass behavior emerges from a common objective and is not an explicit functionality. Nevertheless the velocity of all the individuals will be a variable attribute. This means that the crowd will be slower or faster depending on the speed modeled. We describe our crowd model as follows:

Table 4.2.2: Model attributes description.

Attribute	**Description**
T	time, divided in discretized steps
C	global space
Q	set of states
N	neighbourhood space
β	information related to the grid space N
δ	functions of state transition

$$CM = \langle T, C, Q, N, \beta, \delta \rangle$$

Where the crowd is advancing in discrete times T through a global space C. The movement in the global space C is deterined by the agent states Q, which were defined in Fig. 4.2.1 and Tab. 4.2.1. The space of C considered to take decision by every agent is the neighbourhood N which are 8 cells surrounding the agent's cell. The environmental sub-model will be considering the information of the its independent grid β which implements the intelligence of the agent and the function which will make the agent to move from one state Q_i to Q_j considering the state

variables is δ.

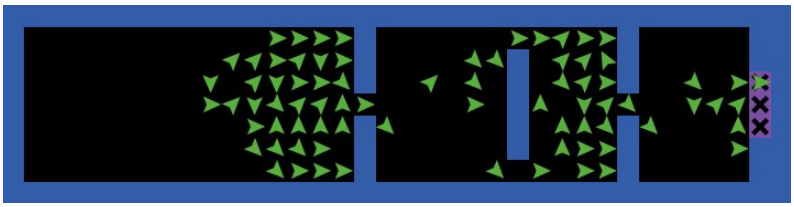

Figure 4.2.2: Discrete model agents navigating.

Fig. 4.2.2 shows the discrete model for a simple scenario. The scenario is the main input of the simulator. It is dependent on the chosen individual model. In our case we have a grid dividing the space that will represent our scenario in a bi-dimensional space. We will name every position of the grid as "patch". Patches occupy a space of 40x40 cm, which is approximately the mean space occupied by a person. The individual analyses the 8 neighbors in random order. As seen in the figure, the agents distribute occupying the space. It tests if the patch has a "free space" type and if there is an individual inside it. Every individual fits a space of the grid, so we will assume the space necessary to hold one person is constant. Because the rules that determine the behavior of the individuals are the same, and also their sizes, the velocity will be the same for all of them. In the space we will find different kinds of patches. They are divided as follows: walls and obstacles: they are the boundaries and the patches that cannot be crossed (even though they are conceptually different in our model, they have the same behavior); free space: available patch where the persons can move (it can be occupied if there is no individual there); exit: objective function of the people (the exit is achieved getting the shortest path).

Algorithm 1 determines the movement of the individuals and the transition of simulation time $\Delta \rightarrow \Delta + 1$. It implements the preciously described model. First, the scenario and space C is loaded in the program. Then, all the agents will be checked in a sequential way, evaluating their neighbourhood and evaluating if it has arrived to the agent and it needs to be released, or if it is approaching the exit

Algorithm 1 Simulation algorithm

```
1: readMap()
2: readExits()
3: generatePotentialField()
4: procedure SimulationStep
5:     e ← targetExit()
6:     for all p ∈ People do
7:         for all n ∈ neighbours(p) do
8:             if n = EXIT then
9:                 releaseAgent(p)
10:            if n = FREE ∧ range(n, e) < range(p, e) then
11:                p ← n
```

and can move.

4.2.2 Environmental sub-model

To get to the exit, we used a minimum distance approach with obstacle avoidance which is based on the gradient of the distance through the goal. We implemented an algorithm to know what the minimum distance which is the "potential field" algorithm. Potential field allows us to provide individuals with a way to be informed if they are approaching the exit. The potential field is based on the idea of the shortest path. Potential field algorithm includes obstacle avoidance and allows to provide us a way to have information about if the agents are approaching the exits. In Fig. 4.2.3 there is shown a map with obstacles and a exit at the top middle of the image. This image shows a color representation of the potential field. Ignoring the dark red colors, which are obstacles. The blue color indicates that the exit is getting closer and the red color that the exit is the farthest. Notice that the obstacles and walls have the darkest red because the agents will avoid these spaces as these are not approaching the exit.

Because the potential field is dependent on the exit, there are as many potential fields as exits. Every agent needs to have assigned one exit, and, therefore, one

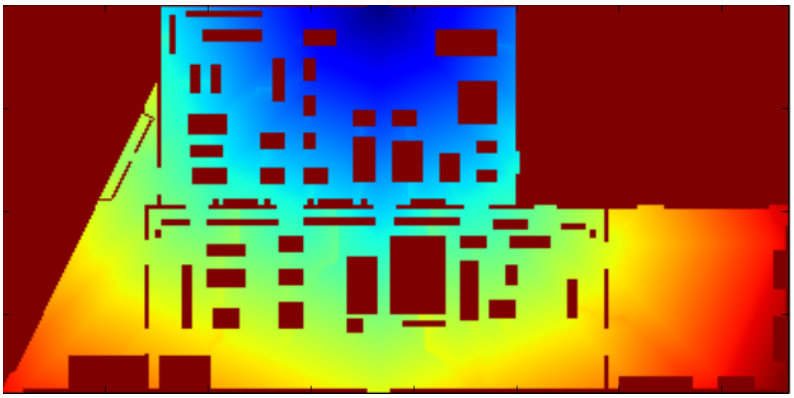

Figure 4.2.3: Potential field colormap.

potential field. The data structure is a list of matrices where every matrix is the flood fill values that are precalculated at the begging and stored in main memory. It is formulated in a discrete manner and the data structure is a list of matrices where every matrix is the flood fill values. There are several potential field algorithms and we chose a mix of the Manhattan and Chessboard potential field calculation[44]. Every individual will have knowledge of the potential field of their exit and will use it to take decisions between steps.

Algorithm 2 Potential field algorithm

```
1:  procedure PotentialField
2:      unvisitedTotal ← 1
3:      pushUnvisited(exit)
4:      while unvisitedTotal ≠ 0 do
5:          for all n ∈ neighbours(p) do
6:              if n ∉ unvisited then
7:                  pushUnvisited(n)
8:                  unvisitedTotal ← unvisitedTotal + 1
9:          if m ∈ unvisited then
10:             calculateFlood(m)
```

The algorithm 2 implements the potential field algorithm. The implementation is recursive, using the implicit stack of recursion to explore the nodes in breadth

way. There is a global structure accessed by all the function calls that tracks the visited nodes and the values are updated with the current value which is updated for every level of the recursion until it fills all the structure. Variables *m* and *n* are temporary variables referring to positions in the map which are being visited.

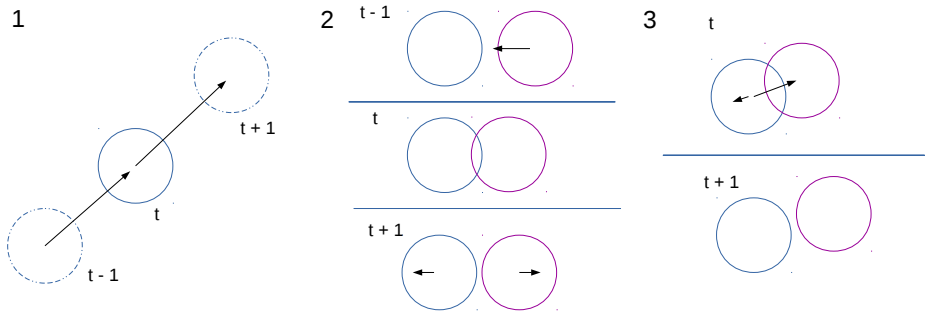

Figure 4.2.4: Continuous time model steps. (1) Acceleration and inertia. (2) Collision. (3) Pushing.

4.3 Continuous time model

In order to reproduce internal behavior of the crowd we had to create a new model which was also based on the simplicity. In our particular case, the continuous model was chosen as an evolution of the discrete model, looking for the ability of reproducing internal of the crowd besides all the functionality that was already implemented. Continuous models can have any position in space and any time because the time step can be adapted as opposite to CA and gain realism in front of discrete models, but the disadvantages in front of discrete models is the computational performance. The neighbors and the environment needs to be checked in every time step. Other approaches are based on stress and panic. We think that psychological and sociological analysis should be included in the model. Our proposal is a model based on psychological and physical factors. The algorithm is composed of three steps that are showed in Fig. 4.2.4. These phases are: acceleration and inertia, collision and pushing, and are detailed in the coming sections.

4.3.1 Agent model

Every agent will contain the attributes that will define it. Tab 4.3.1 describes the attributes used be every agent. Because it is a continuous model there will be an independent position for every agent, and also previous position will be saved and used to calculate the inertia. The radius determines the size of the agent and the vital space and this size will vary depending on the agent which will also have a force to push others. The node coordinates allows to navigate until it reaches this point and then update the

Table 4.3.1: Continuous model attributes description.

Attribute	Description
$\mathbf{r}$	Position of the agent
$\mathbf{r}_n$	Position of the neighbour
$\mathbf{v}$	Current velocity of the agent
r	Pedestrian radius
n	Node coordinates
v_{max}	Maximum velocity of the agent
a	Agent's acceleration
$\mathbf{s}$	Stress threshold
$s\_interval$	Time between push
$\mathbf{f}_f$	Voluntary force factor used by the agent to push
$\mathbf{f}_b$	Backward motion factor after the voluntary push
$\mathbf{f}_i$	Involuntary force factor used by the agent to push
m	Agent's mass

Fig. 4.3.1 shows our model evacuating a closed space. As observed in the figure the agents keep a coherent state preserving the area of their bodies. The figure shows the exiting process is uncoordinated and it produces jams on the areas close to the gate and how they form a clogging. The wall keeps the agents inside the space, stopping their movement and limiting the space. Normally when considering the problem of an evacuation models consider the movement towards a determined point in the space. Moving agents through a complex space involves

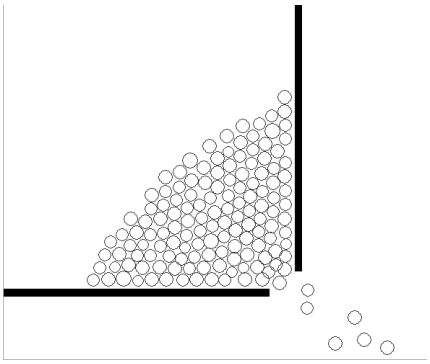

Figure 4.3.1: Continuous model clogging.

knowledge on the agent. Path planning models the movement of the agent through the scenario in interaction with other agents. The agents moves to a certain areas, which are nodes of a graph, and can move in different directions. These areas (nodes) represent the decision points of the agent where there are two or more options on how to continue navigating through the space. Once the agent has reached one of these nodes there is another node assigned to the agent.

4.3.2 Crowd turbulence model

At the beginning of the history of evacuation modeling, some numerical methods where proposed to solve analytically the evacuation problem. Further there was seen they are part of complex system. A numerical method used in game physics and widely extended in computing physics is Verlet integration[38], which implements Newton's equations of motion. The method provides a relatively simple model with good stability.

Collision models allow to propagate forces, which is useful in our case, since the crowd turbulence is the propagation of the interaction between the agents. Fig. 4.3.2 shows our model with the agents accelerating, and therefore, pushing in one direction, the right. It shows how the agents in the wall in the right are more compressed and therefore with more force acting on their bodies, than agents in the left part.

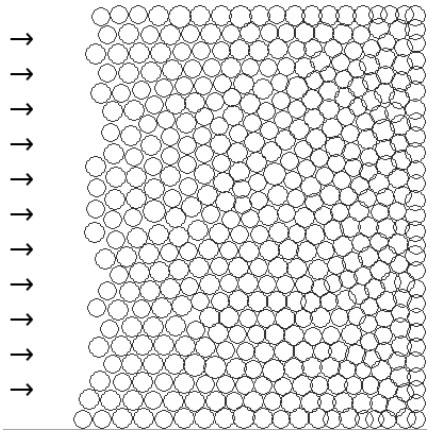

Figure 4.3.2: 441 agents acting in one direction.

4.3.2.1 Inertia and acceleration

Newton's second law of motion describes how the force is produced by an acceleration applied to a mass. We will use this formula as basis to describe the movement of the agents in the space and their behavior as physical bodies. The acceleration allows a stopped body to start movement and reach a determined velocity. Once the desired velocity has been reached we want to keep this velocity and if we stop, we will need to accelerate again.

$$F = m\frac{d^2x}{dt^2} = ma$$

To discretize the equation in differential form we use the representation in finite differences. To transform it we apply the second order central method which defines the division in spaces and neighbors spaces $n-1$ and $n+1$ for different times T.

$$T_1 + T_2 \Rightarrow \frac{\partial^2 u}{\partial x^2} = \frac{u_{n+1} - 2u_n + u_{n-1}}{(\Delta x)^2} + O(\Delta x)^2$$

Applying it to the law of motion we get to the method we get the acceleration.

$$a = \frac{x_{n+1} - 2x_n + x_{n-1}}{\Delta t^2}$$

Isolating the $n + 1$ component we get the calculation of the new x component and we also apply it for the y coordinate getting them in the Verlet Integration form.

$$x_{n+1} = -x_{n-1} + 2x_n + a\Delta t^2$$

$$y_{n+1} = -y_{n-1} + 2y_n + a\Delta t^2$$

These formulas express two phases, the acceleration which is defined by the $a\Delta t^2$, and the inertia which is defined by the first part. We will use this formulas restricting the acceleration to control a maximum speed and keeping always the inertia phase. The acceleration will always create a vector towards the destination node, and the vector will be upgraded once the node has been reached.

4.3.2.2 Collision

The collision is basic to propagate the force of the crowd turbulence. It is necessary to create the wave, move forces trough the mass and accumulate forces. As first instance we considered the formula of elastic collision in a bidimensional space to calculate the physical collision between two bodies. The good part of this formula is that it represents more accurately the conservation of energy and kinetic energy of rigid bodies. But on the other hand, considering people as pure rigid bodies that collide has some disadvantages. This method is applied every time agents intersect and will always keep the vital space and in our case, which is high density situations if the system is consistent agents will not overlap. Also the method in high densities produces jittering which can be solved by reducing the timestep or using techniques to solve this issue. Therefore we could not use this formula. We used the following formula which allows us to have a consistent state in the crowd, have overlapping and produce crowd turbulence.

$$r_1 = r_1 - \frac{m_2}{m_1} d_{ij}(d_{ij} - r_{ij})\mathbf{f}_i$$

$$r_2 = r_2 + \frac{m_1}{m_2} d_{ij}(d_{ij} - r_{ij})\mathbf{f}_i$$

This collision solving procedure will allow to propagate the force providing more realism on the crowd turbulence. It is based on the idea that the force applied is due to an invasion of the personal space, but this can be kept in high density situations. The factors are d_{ij} (distance between agents) and r_{ij} (addition of radius). The distance between two and the radius of agents which will use the distance and invasion of the vital space as factor to reject and collie with other agents. There is a factor applied which will be less than 1, that will make the collision lose energy and reduce the forces. We will use the overlapping as a representation of the pressure applied to the bodies. Also the collision will depend on the mass and lighter bodies will be pushed with more force than heavier bodies. Boundaries define the behavior of the model in the extreme areas of the problem. In the case of evacuation we will consider walls. When colliding with the wall during the simulation, the agent will be returned to a consistent state and it will not bounce against the wall, it will keep the position in the limiting area with the boundaries.

4.3.2.3 Pushing

Pushing is a fundamental part of the model. Other crowd turbulence models do not consider pushing as a specific part but we consider it as important. Pushing is also people losing the equilibrium and falling on others. If there were not people pushing and trying to gain free space, it will mean that agents are passive in high density situations, but this is not what happens. If agents did not push others, it will become stable and stop. Neither are agents constantly pushing, otherwise it will be an unstable system. There is a period of time between two pushes of the same person. As seen in Fig. 4.2.4 (3), one agents push another when it invades its personal space. The agent who is pushing will also have a have a vector force in opposite direction of the way were it is pushing. There is a threshold in the density of the agent where the agent starts pushing. As well as the force of the agent it is a parameter of the model and it is not the same for all. Every time the threshold

is reached the agent will be pushing in time intervals to gain space. Therefore we have the property **s_***interval* which is a simulation time interval and a push factor and backwards factor applied when agents push each other to gain free space.

$$r_1 = r_1 - \frac{m_2}{m_1} d_{ij}(d_{ij} - r_{ij})\mathbf{f}_b$$

$$r_2 = r_2 + \frac{m_1}{m_2} d_{ij}(d_{ij} - r_{ij})\mathbf{f}_f$$

Algorithm 3 Continuous simulation main procedure

```
1:  procedure SIMULATION
2:      for all n ∈ NEIGHBOURS do
3:          if d_ij < VITAL_SPACE then
4:              collide(r_i, r_j)
5:      if isAgentOutside(r) then
6:          r ← calculateValidPosition(r)
7:      r_i ← inertia(r_i)
8:      for all n ∈ NEIGHBOURS do
9:          if isAgentStressed() ∧ stress_time_i then
10:             push(r_i, r_j)
11:     if hasReachedNode(n) then
12:         n ← updateNode()
```

4.3.3 Algorithm

The power of the model is its simplicity and that it is able to reproduce crowd turbulence being easy to parallelize. As opposite to other algorithms it has been thought from the beginning as parallel algorithm. Algorithm 3 describes the behavior of one agent. Because the model is decentralized all the actions can be performed in parallel and the system will keep stable. The algorithm implements the three steps described in the previous section. First, the agent checks its neighbors and detects if there is a collision. If there is, it runs the collision procedure. Then, it

implements the acceleration and the inertia phase moving forward. Finally, if the agent is still not free, and is being compressed, it analyses the threshold and the time from the last push, and it pushes again if they time interval has passed.

4.4 Conclusions

We presented two models in this section. A discrete model based on the division of the space and a second one is an evolution of the first, looking for the more detailed aspects and internal behavior evolving to a definition of a continuous models. Both models correspond to different ways of representing crowd models which have advantages and disadvantages. The discrete model will be computationally more efficient and the continuous model will allow to reproduce internal events of the crowd.

5

Crowd simulation

In this chapter, we introduce the internal aspects of the simulation process as well as the computing model. The simulator has been parallelized and the techniques used are described. The validation and internal analysis of the model are detailed.

5.1 Introduction

The simulator and the implementation of the problem and the application of the theory has some issues related. Considering that it is at the end a piece of software that follows a work flow it needs to have an architecture and design. In our particular case where there is a key importance of the parallelism, this also needs to be described an analyzed. Also the applicability and the implicit needs of the simulator as DSS and the data structures needed for the process.

5.2 Validation

Models are representations of the real system, but we need to prove somehow that our model imitates the reality and we developed a simulator representative and accurate. Validation is concerned with the problem of reproducing models that represent the real system. It is necessary for probing that the model and the simulator are useful for their purpose. In the case of evacuations there are several validation categories with different approaches.

- *Validation against fire drills or people movement.* If the purpose of our simulator is to reproduce the simulation process as a whole system aiming to give information on the total evacuation time we can use real data from experiments with humans and compare the evaluated factors.

- *Validation against validated simulators.* If a simulator has been validated we can use the simulator to compare our results with its results and crosscheck the simulator.

- *Validation for code requirements.* Validation used if a simulator is intended to fit requirements in term of results and is accomplishes this purpose.

- *Validation against evacuation experiments.* Preparing and configuring evacuation experiments helps to understand the flow of the people and internal

phenomena. This is one of the methods that simulators such as as JuPedSim used to be validated[6].

5.3 Discrete time simulator

In the implementation of the discrete time simulator all the information is stored in the data structures. For the discrete time simulator we set up several data structure containing the environment and the agents state variables. The simulator was executed, evaluated lately validated against people movement using an experiment carried out in the Tampere theater (Finland). We also analyse the internal behavior of the crowd in terms of movement and speed.

5.3.1 Data structures

Every agent has a set of state variables that define the transition between the internal states. These variables will define the function to be applied and the transition between time steps.

- *Exit id*: defines the coordinates of the exit and the index to access it will also define the potential field used.
- *Agent id*: defines in a unique way every agent.
- *Coordinates*: position in bidimensional space coordinates (x,y).
- *Released*: boolean variable that tells if the agent has reached the goal and is considered free.

There are two main global data structures which are used to define the environment and the agents.

- *fields*: Array of matrix containing all the potential fields (one per exit). The map data structure will be used at the begginig to create these fields.
- *agents*: Array of structure containing the state variables of all the agents.

There is no data structure that holds the boundaries and obstacles, these are implicit inside the flood structure. An agent will never move to a position with higher higher value in the potential field and move more far from the goal.

5.3.2 Tampere theater

In the particular case of the discrete time model we were interested on the calculation of total evacuation time. There are few public data of crowd evacuation and scenarios with hundreds or thousands of agents. For the functional validation of the discrete time model we used an experiment carried out in 1995, in the Tampere theater[4][79], Finland[74], where 612 people where evacuated. In Fig. 5.3.1 can be seen a the simulation with our model and the agents evacuating using the two existing exits. The black pixels represent the obstacles which in this case are the benches and walls. The stage is located at the bottom of the image and the agents (in blue) move to the green and red pixels which represent the exit of the theater.

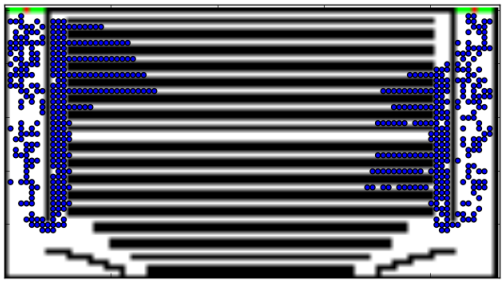

Figure 5.3.1: Evacuation of the Tampere theater.

The mean walking speed for an average person is 1,4m/s[52], but in evacuation cases the crowd speed depends on the density. As described in section 5.3.3 the speed of the mass is slowed down, and to calculate the equivalence of time in our model we used the velocity/density relation based on a linear model to validate our model [47]. To calculate the speed based on the density of the crowd we use the following formulas.

$$v_{crowd} = freePatches * 1,2 + 0,2 \tag{5.1}$$

$$\Delta t = \frac{d_{patch}}{v_{crowd}} \tag{5.2}$$

Where *freePatches* is the mean of the percentage of free patches surrounding the agents, and Δt is the elapsed time in the current simulation time. We selected the linear model being 1.4 m/s the maximum and 0,2 m/s the minimum speed, based on previous studies[47]. Because every agent needs a space of 40x40 cm in our model the maximum density is $6,25 persons/m^2$. The value of a simulation steps varies over the time, where *et* is evacuation time. We get the mean number of neighbours and then divide by 8 to get the percentage. For the simulation the agents where distributed uniformly among all the chairs of the theater.

The evacuation time of the theater was 3 minutes 37 seconds (217 sec) and the people started the evacuation in a time between 25 and 47 seconds after they where informed. The results of the simulator are an evacuation time within the range 205-220 secs using the previous formulas. The simulator assumes that the evacuations starts at time 0. Therefore we can say that the real evacuation time (217 sec) is inside the range given by the simulator (205-220 sec).

5.3.3 Agents speed

The speed of the agents vary all long the evacuation process. In the discrete model even having discrete increases of the position there is a speed of the mass which slows down when nearing the exits. Because of this there can be considered two phases in a evacuation process.

1. *Navigation phase*: agents move from their initial position through the space until they approach the exit.

2. *Clogging phase*: the mass is slowed down due the bottle neck that represents passing a door.

Fig. 5.3.2 shows the agents move in blue color and the evacuated in red. This particular evacuation has 15.000 agents and all of them have been evacuated when

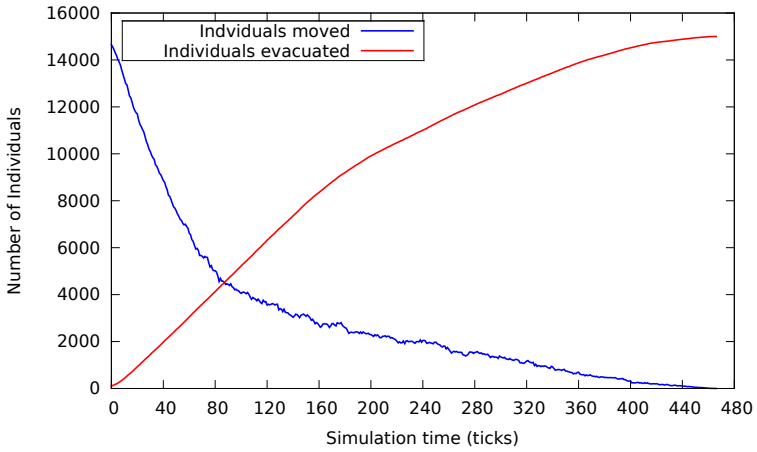

Figure 5.3.2: Evacuated individuals and individuals moved per time

the evacuation finishes. From the graphic can be seen how all the agents start moving when the evacuation starts and we assume this knowledge of them. At simulation time 80, clogging occurs and the people slows down because they are waiting near to the exits. At simulation time 200 the amount of people evacuated per time unit decreases and the evacuation losses intensity, until all of them are evacuated at time 480. This shows how the behavior of the mass is not constant and how the internal phenomenas of the crowd affect to the global behavior of the evacuation.

5.4 Continuous time simulator

The continuous time model represent an improvement and evolution respect the discrete time model including densities, interactions, pressures, propagation of forces, etc. In this section we describe how the algorithm can be optimized, ho crowd turbulence is reproduced and how the data layout is set up. Later the behavior of the model is analysed.

5.4.1 Space partitioning

We recover the idea from the discrete model to optimize: the direct access to the agents. In the discrete time model agents were accessed directly to one position

and also their neighbourhood but in the continuous model there is not such structure to do so. If we want to access the positions of the agents we must iterate over all positions. In the algorithm definition we described how the agent iterates over the neighbourhood, and what we consider by neighbourhood will modify the complexity. If we consider that our neighbourhood is the total number of agents, the computational complexity for the collision phase will be $O(n^2)$. But if we reduce the neighbourhood to a number of agents m, then the complexity will be $O(nm)$ where n is the list of agents and m is the bigger neighbourhood possible.

The elements are inserted always in the first position of the linked list and the rest is moves. So the complexity of insertion is $O(1)$ because the position of the matrix is accessed directly and the insertion is in the first position. Also all the agents in a cell will be accessed in a time $O(1)$ since the position is direct. The data structure is a matrix of linked lists. The matrix is composed by structs which contain an integer variable containing the index and a pointer to a structure. Memory is allocated dynamically in the linked list every time an agent is inserted. We considering other approaches such as KD-tree, but the search and insertion complexity is $O(log(n))$ if the tree is balanced. If the case was not high density, then options such as BSP-trees or KD-trees could be discussed in order to have a better balancing, but not for high density crowds where agents will be concentrated in in the space.

For the size of the cells we have to consider that we have several parts inside of the time step that may modify the position of the agents, so must choose higher value for the cell, considering that the average agent has a diameter of 40cm. Thus, we considered a space of one meter per cell. We can create a smaller size and then have a smaller neighbourhood but we may have inconsistencies, since the data structure is created at the beginning of the simulation step.

Fig. 5.4.1 shows an example of the creation of the data structure. At the left we have agents with their own identifier. The space is divided in a grid of 1 meter cells. Agents may overlap and be even in four cells at the same time but the important variable is the position r. In the example we are considering the agent with id 0,

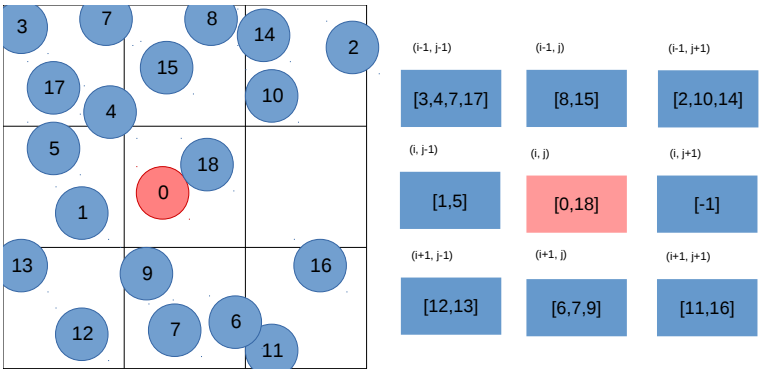

Figure 5.4.1: Example of space partitioning in the continuous model.

in red in the example. The agents are passed to the data structure creating a linked list in every position of the matrix. If there is no agent in one position such as in position $(i, j+1)$, the value of the agent id will be -1, meaning that the list is empty. Agent 0 will iterate over all the agents of all the 9 cells except itself and will compute the collision or the pushing phase of the algorithm.

5.4.2 Simulator structure

The architecture of the simulator is dived by different logical parts. Each part is in charge, and implements a different concept. Dividing the simulator allows us to have a perspective of all the different tasks and concepts that interact during the simulation process.

- *Model*: defines the behavior for every agent and contains the necessary data structures.
- *Simulator*: composes the model and executes the model iterating over time.
- *Environment*: declares the boundaries and the elements that interact with the agents.
- *Logger*: saves the time steps and generates simulation statistics.

5.4.3 Crowd turbulence analysis

The crowd turbulence is complex to analyze due to the existence of only few studies of it. We use the definition of pressure based on previous analysis of real cases to study our continuous simulator.

5.4.3.1 Pressure

There is a lack of data on the case of crowd turbulence. Carrying out experiments to reproduce this event cannot be done due to obvious ethical reasons. It is hard to reproduce by video analysis or other techniques and exact turbulence, but on the other hand the approach has been done reproducing the global behavior of the mass. To validate the ability of reproducing crowd turbulence we use the definition of "pressure" proposed by Helbing. The formula has been used in the study of the crowd turbulence in the Hajj, Mina, carried out by Helbing[31]. It was based on the video analysis of the Hajj disaster with 3 million pilgrims in Mina. Later, the formula proposed was used to validate an extension of the social force model[82] which reproduced the turbulence as well as the internal movements and velocities of the agents inside the crowd. It was also used by Golas[25] to validate its proposed continuum model. The formula for "pressure" that depends on the variance of velocity and the density. When the values for a certain region is higher than 0.02 it assumes that there is a crowd turbulence happening in a local area. The formula is described below.

$$p = \rho_i Var(\vec{v_i})$$

$$\rho_i = \sum_j \frac{1}{\pi R^2} exp\left(\frac{-\|\vec{r_j} - r_i(t)\|^2}{R^2}\right)$$

This formula reflects the main aspects affecting the creation of the crowd turbulence which are the density of the agents, the distance to a local agent and also the sudden change of velocity of the mass. The value p is calculated for an agent i in a region and all the neighbours j are evaluated. R is declared as a parameter and

we will use as value the mean diameter of an agent. We used the diameter of the local agent we are considering as parameter of R. Image 5.4.2 shows the evolution of the ”pressure” for a certain area as simulation time evolves using our model. We can see how at time 8 the pressure starts increasing, and at time 10 the ”pressure” exceeds the threshold of 0.02 continuing with the increase. At the beginning the crowd is stopped and then it starts increasing the movement of the turbulence leading to a higher compression. Due this abrupt behavior, some descriptions made an analogy with earthquake[31].

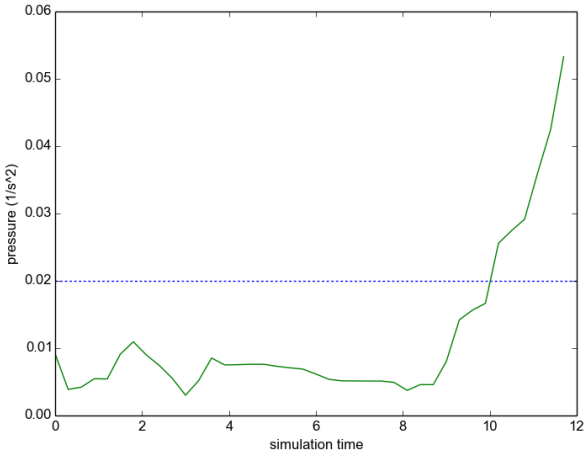

Figure 5.4.2: Reproduction of a crowd turbulence.

On the other hand we also wanted to visualize the evolution of the pressure of a crowd turbulence. For this we created a heat map representing the pressure in a certain region of a high density area. Fig. 5.4.3 represents this timeline where the pressure evolves every frame. This image shows how the agents in the center of the wave are the ones supporting higher pressure.

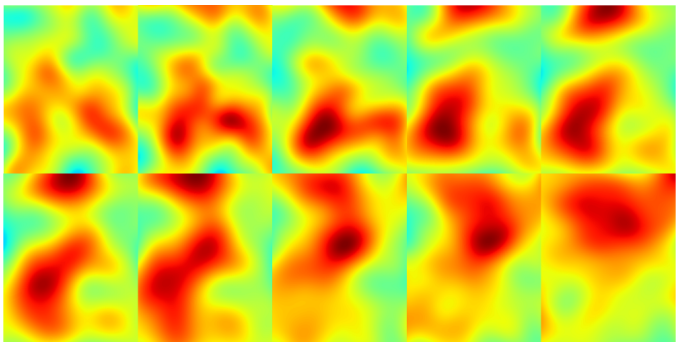

Figure 5.4.3: Frames representing the evolution of pressure of a crowd turbulence from left to right and from up to down.

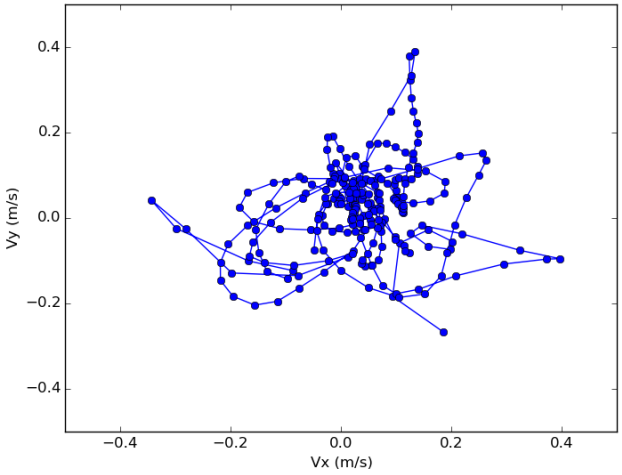

Figure 5.4.4: Agent's velocity evolving in a high dense crowd.

5.4.3.2 Agent movement

When an agent is inside a high density crowd the movement is not linear. At times it moves to the exit, other times it is slowed down and stopped, others is moved backwards due the movement of the mass. Fig. 5.4.4 depicts the velocity over v_x and v_y of a pedestrian using our simulator. The velocity evolves going from completely stopped to movements against its desired will, where the agent is moved by the mass. This shows a random pattern determined by the crowd.

5.5 Statistical reliable results

In evacuations, and particularly in crowd evacuations there are random patterns among the public. This will mean variance and difference on the results. Our perspective is that a lot of stochastic factors impact on the model. Reproducing an accident with one simulation can be hard. But that that is because a lot of factors impact on the simulation. That is why we need a statistical significant amount of simulations, in order to have a set of data that will allow us to have enough results, and more reliable. We will use statistical methods to calculate the number or runs based on the statistical reliability of the results or the variance. For our work we proposed two methods: Diaz-Emparanza and Chebyshev's inequality.

Also there can be several configurations. For example if there are two exits we can test how the scenario will behave with 50% of the people going to each exit, or testing how it will be to make all the people go only to one. The simulator aims to provide knowledge about the possibilities and the configurations. Because of this, the combinations of configurations of the scenario are necessary to provide information.

5.5.1 Diaz-Emparanza

Diaz-Emparanza[12] proposed a method based on probability intervals and accuracy to determine the minimum number of runs necessary. Considering the simulation total execution time, we can separate simulations between terminating and non-terminating, which depends on convergence and if there is a final condition for the simulation process. In evacuation the simulation is terminating because in general the condition is that all the agents have been evacuated. Once focused this point, the formula of the method can be used.

$$T = \frac{t^2_{\frac{\alpha}{2}} p_H(1 - p_H)}{A^2} \tag{5.3}$$

Where T is the number of simulations, in our case we use the following values with which we get the minimum number of simulations. Our probability interval

is [0.045 0.055]. As example we selected the following values:probability interval (p_H): 0.05; and accuracy (A): 0.002; level of confidence ($1 - a$) where we get Student's t-distribution value.

$$T \leq \frac{2.3^2 0.05(1 - 0.05)}{0.002^2} = 62,818.75$$

Before we analysed the simulation execution time for 1,000 independent runs and the results approach a normal distribution testing with chi^2 and Kolmogorov-Smirnov method. Then we can use the previously calculated value as the minimum number of runs that we need to meet the previous criteria.

5.5.2 Chebyshev's inequality

As compliment to Diaz-Emparanza method we also use Chebyshev method. It allows us to calculate the number of runs in function of the variance. We use the formula of Chebyshev's inequality generating N simulations, calculating the standard deviation, and then calculating the % of the data values of the data distribution. The Chebyshev formula is the described as follows.

$$Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

Where X is a random variable, μ the expected value, k any real number, and σ the variance. Within the given percentage it needs to be manually calculated which is the amount of data necessary. Meanwhile Diaz-Emparanza is used for the calculation of the total number of runs, this method is used to calculate the number of simulations for different parameters of the configuration of the simulation. In that way we can launch different simulations with different configurations and know the total amount of them that we should do.

5.6 Scheduler and configuration

The scheduler aims to distribute the independent simulation among the nodes of the cluster. It communicates using the network with other nodes initializing the

simulation with a determined configuration. Fig. 5.6.1 depicts the scheduler and how the master is giving configuration to nodes than can be composed by multi-core architectures and GPUs.

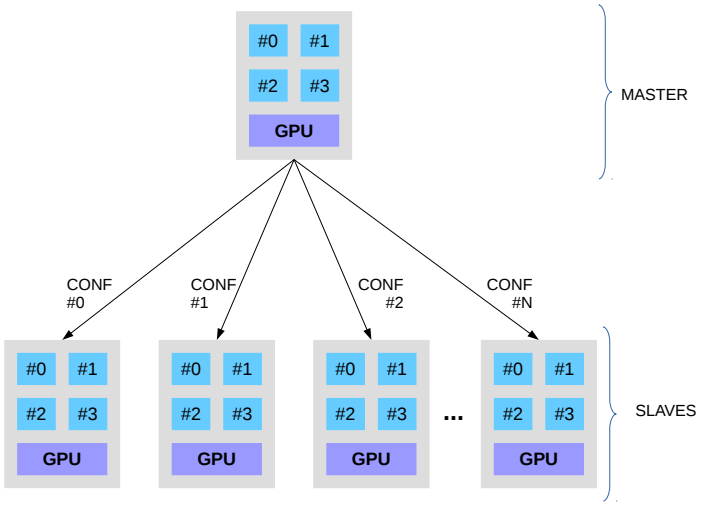

Figure 5.6.1: Scheduler scheme.

The configuration of the simulation is known by the master. It will receive the configuration of the model (discrete or continuous) and the attributes configuration, scenario, number of agents, etc. The number of simulations is divided among the total number of nodes and then distributed using Diaz-Emparanza based on the total number of simulation or Chebyshev based on the variance of the input parameters of the simulation and calculating which percentage of the results is within a mean and standard deviation. The result of every simulation is unique and different. Because of this, all experiments are independents and a scheduler is responsible to run the compute on idle nodes. This is embarrassingly parallelism, which allows to run experiments on isolated nodes. We used the MPI simulator developed for the discrete model. The simulation will be stored with unique IDs in a parallel file system in order to be analysed in a post analysis process.

5.7 Conclusions

We have seen the internal aspects and design of the discrete and continuous simulator. When developing it there are some theoretical decisions than need to be taken and a posterior analysis to check if the results that the simulation is producing are correct. Moreover we have validated that our continuous model is able to reproduce crowd turbulence by using the "pressure" formula which was generated by visualization of crowd turbulence disasters. A computational optimization for the continuous algorithm has been discussed, where the idea of efficiency of the discrete model is reused to optimize the performance.

6

Experimental evaluation

This chapter shows, the experiments prepared and developed for the discrete and continuous models for different scenarios and configurations. It also discusses the performance and parallelization results for different parallel programming models.

6.1 Introduction

During the evolution of the models, we wanted to evaluate several aspects and test these against real cases and synthetic scenarios. First of all we evaluate the discrete time computational needs with an extended ABM simulation tool, and then we evaluate our model with HPC architectures. Once it has been tested in a cluster we evaluate how it behaves as a DSS on a Cloud server to provide a tools to the experts and our workflow. We also test the continuous model on GPU architectures using synthetic scenarios comparing it with the same algorithm for a sequential algorithm.

6.2 Discrete time simulation

We present performance results for the discrete time simulator. To evaluate the performance we used several real scenarios that could handle thousands of agents. After that we digitalized the scenario and then we configured and ran the simulations dividing the simulation with the scheduler.

6.2.1 Manga festival

The event *Salo del Manga*[63] is an annual meeting taking place in Barcelona since 1995. It hosted 105.000 visitors in 2013 and this big attendance motivated the election of this space for our testing. We chose the building 2 of *Fira de Barcelona* [22] to test in first instance our discrete time simulator. This space receives international conferences and fairs with thousands of attendants. The aim of this experiment is to first evaluate the performance needs of current simulation tools and later, evaluate the performance of our scheduler and simulator.

6.2.1.1 Scenary

In the Fig. 6.2.1 can be seen the original map. We did simplifications in the map digitization involved chairs, platform, importance of exits and the bar exit. We

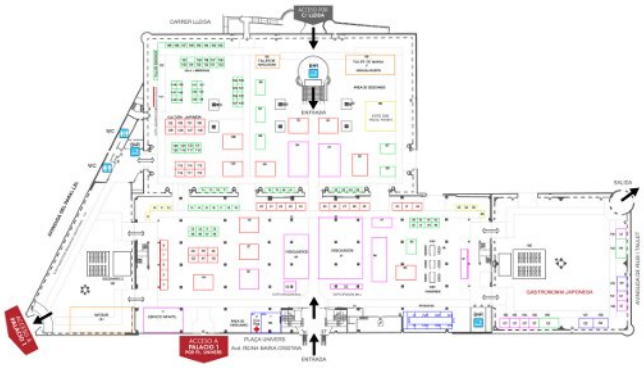

Figure 6.2.1: Map of *Fira de Barcelona*[49]

simulated the space *Fira de Barcelona* with populations of 3,000, 6,000, 12,000 and 15,000 individuals which were randomly distributed among the space before the evacuation started. Because we wanted to evaluate the performance needs of current tools we implemented the model in the tool NetLogo and later in the simulator able to distribute the simulation with OpenMPI in a MPI cluster.

Figure 6.2.2: Fira del manga scenario simulated with NetLogo

6.2.1.2 Performance NetLogo

We developed the model described in section 4 using NetLogo[80], a widely applied modeling tool for ABM programming and an integrated modeling environment. NetLogo is commonly used in research and academia allowing the analysis of emergent behavior for different areas. We wanted to study the behavior of the

model and also analyze the response time of the simulator as a widely used tool in ABM simulation. We digitalized the scenario which can be seen in Fig.6.2.2 using as reference image 6.2.1. We used the same exits and maintained the stands as obstacles. In the simulation agents were distributed randomly at the beginning of the simulation. In NetLogo we implemented the behavior using Logo language and we create a breed of people to define persons. The space in NetLogo is divided by patches and we use a path to host one agent defining procedures to define the behavior of the agent. Fig.6.2.2 shows how agents are moving towards the exit occupying one path per agent and avoiding obstacles. Clogging is observed on exits and how the mass slows down waiting for other people to evacuate.

Table 6.2.1: Node NetLogo characteristics.

Component	Characteristics
CPU	Intel Core i5-2400 CPU 3.10GHz
Cache	6MB L2 (2x2)
RAM	8GB
Java	Version 1.6.0_35

We run the NetLogo simulator with the configuration described at Tab. 6.2.1. In Fig. 6.2.3 can be seen the execution times of the NetLogo model which are the means of 100 runs. This shows how the the simulation time grow as the number of individuals increase in NetLogo. In the figure, the tendency of execution time in a tool such as NetLogo shows how time is increasing in a tendency above the linearity for a linear increase of the number of agents for a discrete time model. In more complex models such as continuous time, the tendency will raise with the increase of agents. If we follow the tendency, for a case where we want to simulate 150,000 agents it will take near to 1,740 sec, around 30 minutes for only one simulation.

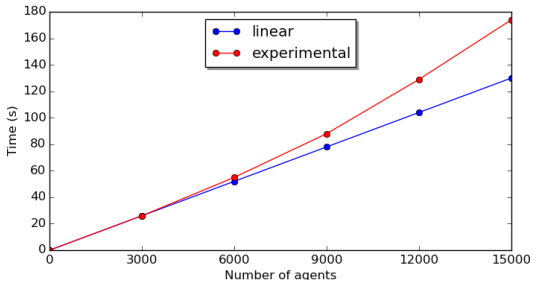

Figure 6.2.3: NetLogo computing time for different number of agents.

6.2.1.3 Performance MPI

Once we evaluated the performance of the NetLogo simulator we implemented a version for the same discrete time model but implemented with C distributed with MPI. There is no multithread parallelization, the batch system maps one process per core to increase the usage of the machine. We used the same digitalization of the scenario which is showed in Fig. 6.2.4. The simulation time selected to plot the evacuation instant is similar to the selected in NetLogo Fig. 6.2.2 to evaluate visually that the behavior is similar.

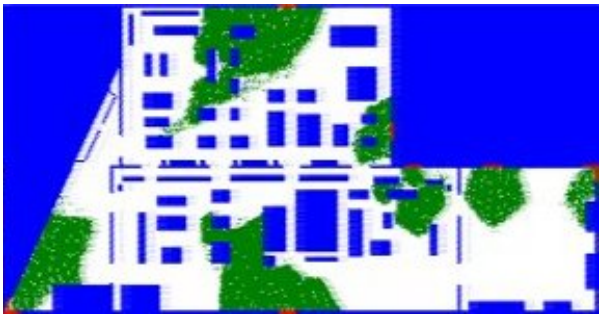

Figure 6.2.4: Fira del manga scenario simulated with MPI

The characteristics of the cluster used are specified in Tab 6.2.2, the number of executions selected is 62,820 runs, which is the number of simulations calculated

Table 6.2.2: Dell cluster characteristics.

Cluster	Characteristics
Nodes	32 IBM x3550
CPU architecture	2 x Dual-Core Intel(R) Xeon(R)
CPU	CPU 5160 @ 3GHz 4MB L2 (2x2)
Memory	12 GB Fully Buffered DIMM 667 MHz
Disk	Hot-swap SAS Controller 160GB SATA Disk
IO	Integrated dual Gigabit Ethernet
Compiler	GCC 4.4.7
MPI library	OpenMPI 1.6.4
Parallel FS	NFS

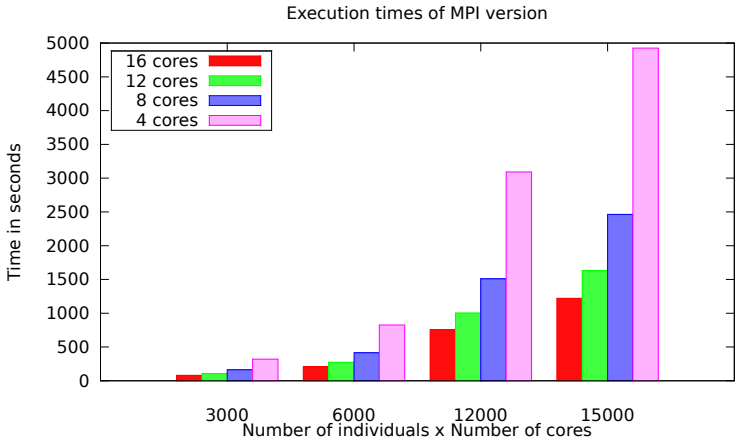

Figure 6.2.5: Distributed simulator execution times

with Diaz-Emparanza for an accuracy of 0.002 and a probability interval of 0.05. This number of runs is a parameter of the scheduler which will automatically manage the simulation among all the workers. Because in the scenario there are a total of seven exits, there are seven potential field maps, one per exit. The other data structures are stored in arrays or array of structures. In the Fig. 6.2.5 can be seen the execution times of this simulator for 4, 8, 12 and 16 MPI. The time decreases significantly due to the increase of nodes thanks to the MPI distribution of the executions. We also executed the serial version for 62,820 simulations and compared

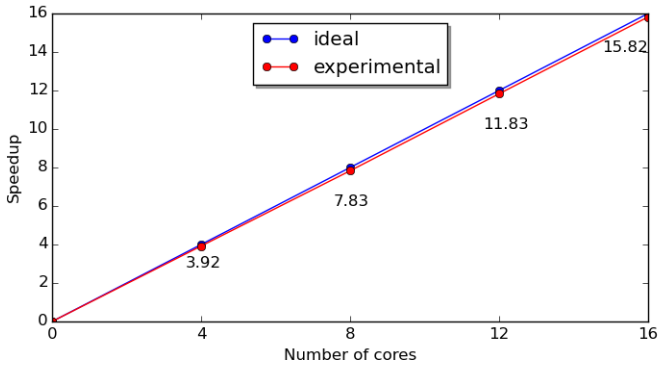

Figure 6.2.6: Speedup Manga scenario.

the result time with the parallel versions. In the graphics seen in the Fig. 6.2.6 the speedup showed becomes linear. As the efficiency is close to 1 it can be said that the scheduler scales and is efficient. The differences in the scalability remains in the variances of the simulations with a stochastic component.

6.2.2 Palio di Siena

The second scenario we selected to test our discrete time simulator was *Palio di Siena*[73]a, where around twenty thousand persons meet to see an event[55]: the horse race. The aim of this experiment was to test an outdoor event with a higher number of population on a cloud architecture and test our DSS workflow.

6.2.2.1 Scenario

As shown in Fig. 6.2.7 (a) the *Piazza del campo* is a closed space with several exists and during the event *Palio di Siena* which is a historical horse race that has its origins on the XV century, where the people concentrates in the center of the square to assist the show. In this case we did not select a random position for the crowd, instead we set up the agents in the center as seen in the Fig. 6.2.7 (b) with the scenario digitalized because is where the agents stand observing the race. The amount of agents used for the simulation are 28,378 persons filling the center of the square. In the figure can be seen how there is a black space representing an ob-

stacles in the top center of the blue area, which is *Fonte di Gaia*; the white spaces, which represent the free space; and the blue patches, which are free spaces filled by people.

Figure 6.2.7: (a) Aerial view of the *Palio di Siena*[11]. (b)Siena scenario input map.

6.2.2.2 Performance

For the simulation we used cluster described in Tab. 6.2.3. To select the appropriate Amazon instance we first analyzed the computational requirements of our simulator. We used the *perf* tool to analyze the memory usage with the case that will be used in the experimental part. The execution is waiting 27.1 % frontend idle, 12.04% back-end idle. In the Siena case, the program mallocs dynamically 6.6MB of data. In this case, we have a mean of 1,950 page faults, where every memory page is 4 KB. This means an average access of 8MB to main memory and that our problem may be limited by memory. Choosing a more powerful CPU instance will waste resources and money. Also, the memory described in Tab. 6.2.3 is necessary to store the data structures, the software dependencies for our simulator and the operating system. That is why we are working with the previous instances. We selected several instances and numbers of nodes (4, 8 and 12) to test the performance of this parallelization. In this case, we use the local drive to store the partial results. We use the local file system to store temporary results. As we see in Fig. 6.2.8 the speed-up show to be almost linear for all kinds of instances, and this

Table 6.2.3: Instance m1.small cluster characteristics.

Cluster	Characteristics
Nodes	1-12 EC2 instances
CPU	1.2 GHz
Cache	4MB L3
Memory	613MB
Compiler	GCC 4.7
MPI library	OpenMPI 1.6.4
Parallel FS	NFS

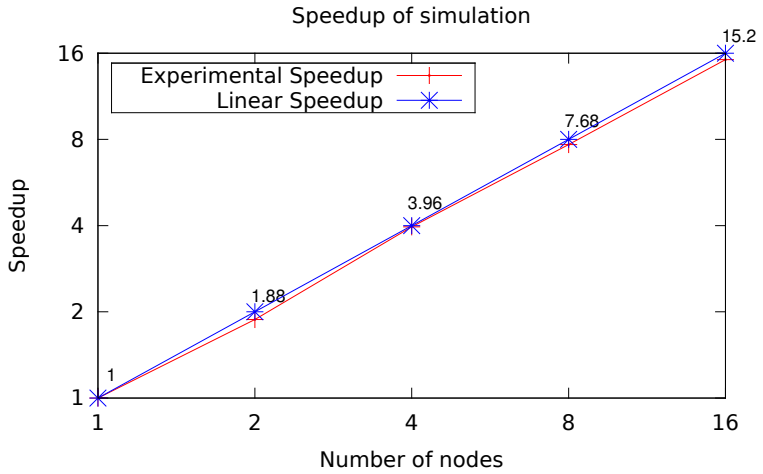

Figure 6.2.8: Siena scenario speedup

means the application scales in the infrastructure. This is due to our parallelism.

6.2.2.3 DSS and Cloud

The aim of a simulator is to provide knowledge and contribute to the analysis of crowds for experts and use the simulator as a DSS (Decision Support system). To hide all this complexity we will move the simulator to the cloud, and particularly to the Amazon infrastructure. Under the need of having an MPI cluster on demand we used the project named StarCluster[51]. StarCluster is an open source cluster-computing toolkit for Amazon's Elastic Compute Cloud. To execute it on

the cloud and make it SaaS, we create a system execution flow from a web interface until the output of the system. The cloud resources intercommunication is hided by StarCluster. It establishes SSH connections between nodes and creates an MPI cluster.

In order to execute the simulator using a web interface from the cloud, we created a whole execution flow to run the simulator on the cloud. The web page is responsible for collecting the input data and returning the output as it is showed in Fig. 6.2.9. The necessary input for the system will be: bi-dimensional map which may optionally include the assistants. We created a web page that uploads the map and the rest are introduced using a online form which transfers the data to amazon. Once all this information considered mandatory to run the model is set, then we assert all the input is correct. As input of the simulator, a loss-less image compression format should be used because every pixel will contain information about one part of the map. In our case we chose the PNG format. The image can be created using satellite images, city/building maps, etc. Every pixel represents information about the bidimensional space and the scale of every pixel is 40x40 cm^2. An example of input image would be the Siena map input showed in Fig. 6.2.7 (b) where the information of the pixels what we described previously.

The StarCluster toolkit[51] automatically configures the cluster but it must be configured beforehand. Once the simulation is prepared to start running, a Python script launches the OpenMPI job to the queue system. To distribute the simulations among all the nodes, a static scheduler is used. All the nodes get a fixed number of simulations. The nodes save the information locally in a NFS file system. This minimizes the communication among nodes and makes an independent approach where there is no dependency. Once the simulations have finished, the user will have feedback with a report that will be sent to its personal e-mail. There will be a report showing information such as: mean, minimum and maximum evacuation time, people evacuated per gate, and a 3D visualization of the evacuation. This interface will be a tridimensional interaction with the bidimensional execution, allowing another axe to navigate. The output and interaction of the system

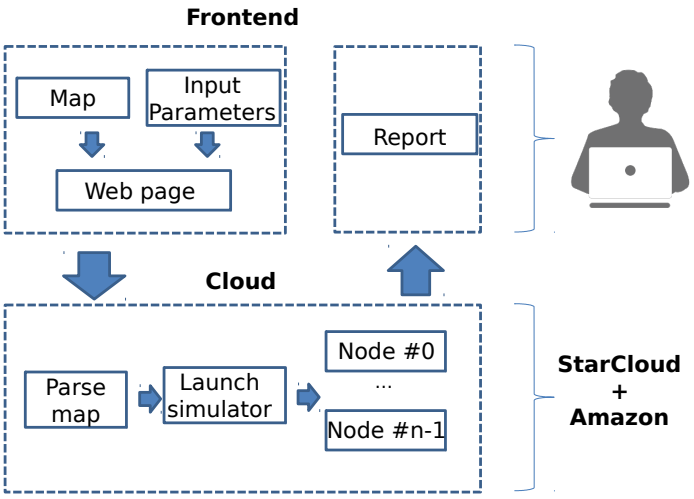

Figure 6.2.9: Cloud workflow.

are crucial. The user experience can determine if a system is used as well as any technical quality. For the visualization, we used WebGL and we reproduce a simulation on the web browser. The user can browse time and space, going backwards and forward. To implement the workflow, we implemented the front-end using the Apache web server with a Python module to program the web scripts.

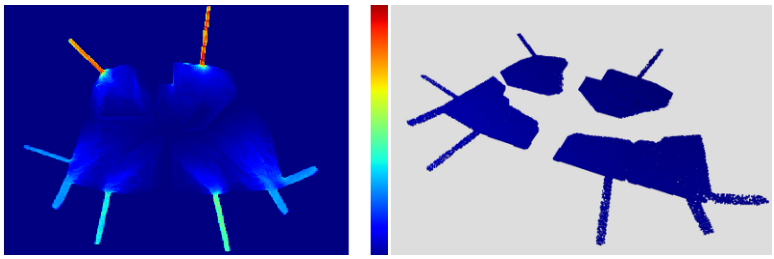

Figure 6.2.10: Siena scenario output

The performance of the tool is one of the most important factors, due the fact that the results of the simulator is the aim of the present research. We also calculated the heat map in order to analyze the zones where there is more concentration

of the public, and also more possibilities of accidents. In Fig. 6.2.10 (a) we can see an example of a heat map with the viewer of the system. In the Fig. 6.2.10 we can see an example of the crowd simulation viewer. In this case we have a simulation step where we can analyze agent's positions and the distribution among the space.

6.3 Continuous time simulator

We implemented the continuous time model into a simulator and we evaluated its performance using several scenarios. We performed two experiments: one in the GPU simulator using agent division; another using space partitioning in a CPU. The GPU version was compared with the performance of a sequential version to compare the results and the partitioned version with the original.

6.3.1 GPU parallelization

To first test our model we created a synthetic scenario. We selected this option, because using a synthetic scenario allows to adapt our simulator and test different number of agents in order to test the performance over the number of agents maintaining the same density, in our case 6. Therefore, the size of the boundaries are changed as the populations to be tested vary. The area, shown in Fig.6.3.1 is a closed rectangular space with two doors where the agents are evacuated.

In the continuous model the simulation had to be distributed among the computing threads. To parallelize for SIMD architectures we did not divide by space because this could lead to load unbalancing and thread divergence in the GPU. Some cells could be empty will other could be overcrowded. Instead of this we parallelized by agent. Every thread owns an agent and performs its behavior for a time step. Every agent executes the kernel described in section 4.3.3. The thread will read the position from other agents using a common structure stored in shared memory.

The data layout wants to find an good configuration of the data structures in memory in order to improve the performance taking into account the computer

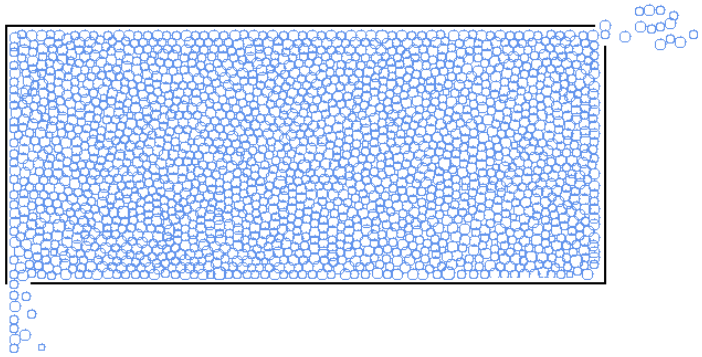

Figure 6.3.1: Rectangular area continuous model evacuation.

architecture and the memory access pattern of the algorithm. The data structure to parallelize in the GPU is a SoA (Structure of Arrays). This is due the way the GPU access the threads and the memory access pattern. One of the main problems is threads divergence. The SM share the program counter, and the GPU is good when all the cores are doing the same inside the same SM. The problem is that the collision phase is not applied to all the agents at the same time, it is only calculated with intersecting agents.

6.3.1.1 Performance

We executed the simulator on the GPU and we also executed a sequential version on a CPU to compare the performance with the parallel version. The experimentation platform is described in Tab. 6.3.1. The space geometry is saved in memory. They are created in the host and transferred to the GPU at the beginning. During the simulation, every simulation step, coordinates are transferred from the device to the host in order to observe or save the evolution of the crowd.

GPU performance was compared with the sequential version executed in a CPU. In Fig.6.3.2 we display the results of our experiment. We compare 1.000 simulation steps for different numbers of agents on the GPU version. The per-

Table 6.3.1: Node GPU GTX 750 characteristcs.

Component	Characteristics
CPU	Intel CPU with 2.1 GHz
Cache	12MB L2
RAM	8GB
GPU	Nvidia 750 GTX, 1.1GHz 2GB memory
Compiler	GCC 4.4.7
OpenCL	Version 1.2
GCC flags	O3

formance of both versions doubling the population between 512 and 4096 agent. The time is scaled in logarithmic scale. Increasing the number of agents the performance is comparatively improved in the GPU version making usage of idle SM and making a more efficient usage of the resources. In the GPU case, the running time includes memory transaction from the GPU to the host with the coordinates of all the agents in order to save the step. The data movements using the PCI connection penalizes the GPU version, where it stores locally the data. As we increase the number of agents in a logarithmic fashion, there is no significant performance penalty. This means that it is efficiently scaling. The data parallelization and the total independence on write operation are showed in these results. Each agent is working and manipulating only their own positions and reading everybody else position.

Fig. 6.3.3 shows the simulator executing for 8 iterations of the algorithm executed in a M2090 NVIDIA GPU card. The image shows how there is a transfer of the positions in every iteration from the GPU to the host. This data transfer is then stored in the local file system.

6.3.2 Space partitioning

As second test we implemented the space partitioning algorithm described previously to evaluate its performance. In the simulation all agents have different goals

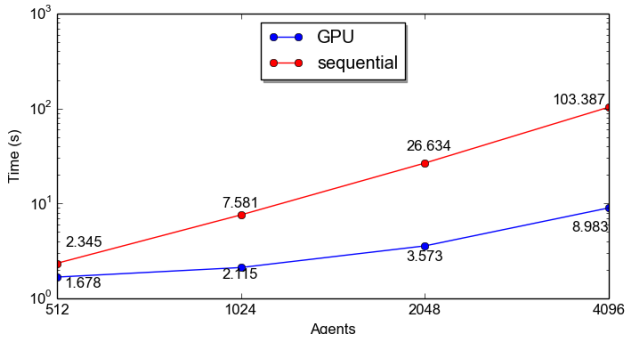

Figure 6.3.2: Execution time continuous model GPU and sequential for 1,000 iterations.

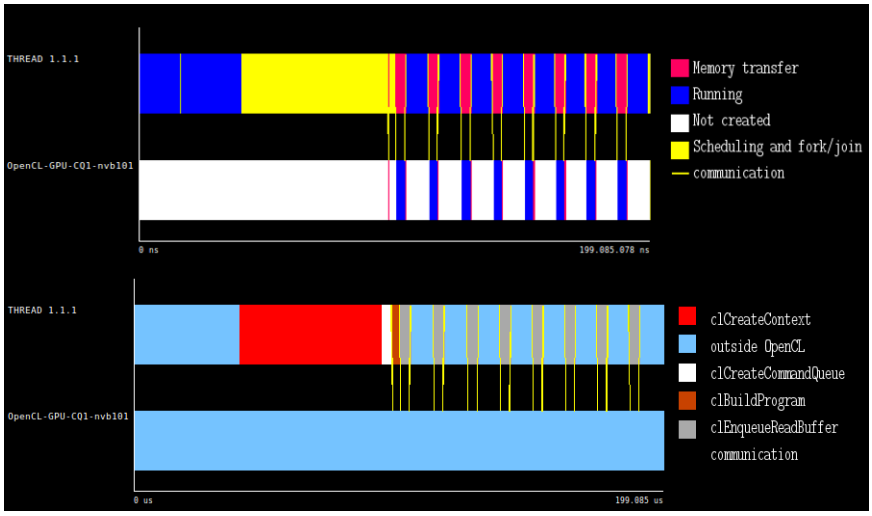

Figure 6.3.3: Analysis of the OpenCL actions. (a) Memory, scheduling and CPU operations. (b) OpenCL calls in host.

and move through the space in different directions. The space is a square room which keeps an average density of around 6 agents per m^2. Fig. 6.3.4 shows an example of the scenario for 15,000 agents. The size of the cell in the grid division is 1 meter. There are two parts with quadratic complexity: the collision phase and the pushing phase. We apply the technique to both. Table 6.3.2 describes the system used to evaluate our algorithms.

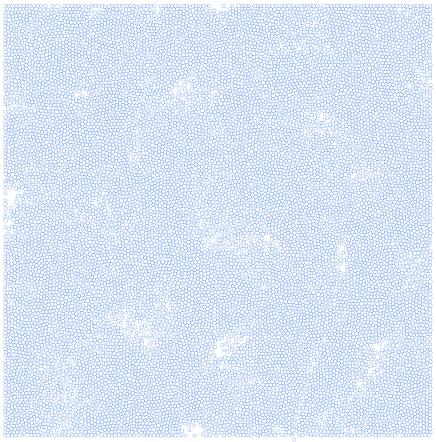

Figure 6.3.4: Scenario with 15,000 agents.

Table 6.3.2: CPU characteristics for partitioned version.

Component	Characteristics
CPU	Intel i7-4600U CPU 2.10GHz 4-cores
Cache	L1 (64KB), L2 (256KB), L3(4MB)
RAM	8GB
Compiler	GCC 4.9.2
GCC flags	O3

6.3.2.1 Performance

We performed several experiments for populations from 10,000 to 40,000 agents adapting the size of the scenario, and therefore, modifying the size of both data structures: agents and space partitioning structure. In Fig. 6.3.5 we show the execution times for the original version and the partitioned version in logarithmic scale for 1,000 simulation steps. As seen in the graphic, reducing the complexity has a direct impact on the performance of the algorithm. There is an overhead created every iteration by the creation f the data structure, and that there is a bigger amount of data accessed, increasing the total amount of memory consumed, and

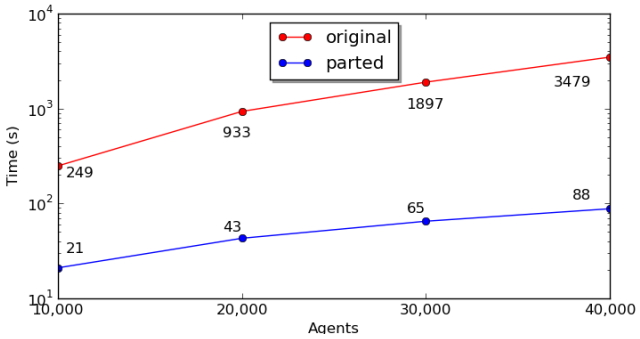

Figure 6.3.5: Execution time original and parted version 1,000 iterations.

the reduction of memory accesses improves the performance.

In Tables 6.3.3 and 6.3.4 we analysed the computational metrics to analyze the results of the optimization. When analyzing the instruction per cycle (IPC), we see how in the original version it was more efficient and there was a higher instruction level parallelism. The memory hierarchy was used in an efficient way, due that the data was accessed in a stride-1 because the outer and inner loop of the comparison phase were incrementing the ids sequentially. On the other hand the number of cache misses in the partitioned space version is higher because there is more space used in memory due to the creation of the grid structure, and also the access pattern is stride-1 for the outer loop, but the inner loop accesses only to the neighbors of the current id. Nevertheless the usage of the memory hierarchy in the partitioned version is quite efficient, even that for higher number of agents and spaces the cache misses decreases. The most important and determinant factor in the performance of the algorithm are the total reference to cache. The quadratic complexity version is accessing in big ranges that penalize the global performance of the algorithm. In the second version, where the behavior of the simulator is not penalized, the number of accesses is dramatically reduced, where even in the case of 40,000 agents in the partitioned version it has less cache references than the original version with 10,000 agents.

Table 6.3.3: Original version performance.

Agents	Scenario	IPC	Cache misses	Cache ref
40,000	8,200x8,200m	2.93	0.26%	106,200M
30,000	7,100x7,100m	3.05	0.16%	59,025M
20,000	5,800x5,800m	3.06	0.16%	24,606M
10,000	4,100x4,100m	3.05	0.38%	3,796M

Table 6.3.4: Partitioned space performance.

Agents	Scenario	IPC	Cache misses	Cache ref
40,000	8,200x8,200m	1.33	3.71%	2,674M
30,000	7,100x7,100m	1.34	3.1%	1,975M
20,000	5,800x5,800m	1.36	2.5%	1,218M
10,000	4,100x4,100m	1.39	1.34%	460M

6.4 Conclusions

We described several experiments using the discrete and continuous time simulators using different scenarios to evaluate their performance. The parallelism and performance were evaluated on different architectures and discussed. The discrete model was implemented and distributed in a cluster, dividing the total number of simulations. We also created a workflow for evacuation simulators to provide an easy interface to execute HPC evacuation simulation on the cloud, and we demonstrated how the system scales in these architectures and the simplicity of our workflow. Later, the continuous model was implemented in a GPU and divided by agent parallelization, increasing the average performance over the sequential version. The division by agent and the lack of dependency makes it suitable for this kind or architectures. Then, a space partitioning technique was applied to the algorithm demonstrating how it speedups the average performance of the simulation.

7

Conclusions and future work

This chapter exposes, several concluding thoughts on the work exposed showing the contributions of this thesis listing the research papers developed. Finally it opens new research lines analyzing the future of this work.

7.1 Concluding remarks

Crowd evacuation simulations can take advantage of high performance architectures. These simulations are complex, involving thousands of occupants in a determined space. Also non deterministic behavior produces a variance in the result and also there is a variance in the setup of the sociological and psychological components of the population of the evacuation. These executions will have a big cost in sequential environments, and it will be penalized as we increase the population of the crowd. Moreover, modeling human behavior is a complex task, handling thousands of agents can be a demanding task. When reproducing internal phenomena that happens inside the crowd which are internal aspects of the reality the system becomes more complex and more factors and variables are involved in this complicate the model. A crowd model need to show stability and needs to reproduce the behavior. In the particular case of crowd there is few data and crowd turbulence is analyzed and validated through formulas extracted from the analysis of real disaster. Once the model and the simulator have been created it needs to be prepared to allow the usage to final users using a DSS which can take advantage of the cloud. We have developed these ideas and the contributions developed can be summarized as follows:

- A discrete time model has been developed and implemented. First it was implemented in a ABM simulator (NetLogo) to evaluate its performance on this kind of tools and the computational needs.
- We performed several experiment using the discrete time simulator with several real scenarios, evaluating the performance of the simulation.
- The discrete time model results have been compared to experimental results carried out in the Tampere theater to validate the egress time performance for this case.
- We developed an evacuation DSS that runs on the cloud, and follows a simple workflow, making transparent the model, the performance and the in-

frastructure to users of evacuation simulators.

- A crowd turbulence model has been developed based on Newton's second law of motion and discretized using the finite difference method. The model was complemented with collision algorithm and psychological factors induced by the pressure.
- The crowd turbulence has been parallelized for SIMD architectures and it has been designed from scratch to follow this pattern. It has been executed on GPUs and its performance has been evaluated showing a speedup respect to sequential versions.
- A data layout has been set up to allow agents parallelize, and increase performance on GPU architecture.
- A scheduler has been developed following a configuration determined by the Chebyshev's inequality and also for Diaz-Emparanza method. This scheduler follows a master worker paradigm allowing to have statistical significant results, as a approach to the problem that its difficult to reproduce an evacuation situation.
- It has been showed experimentally how the crowd turbulence model has been able to reproduce turbulence using the definition of "pressure" to demonstrate it and also the movement of agents and displacement of the mass.

7.2 Future directions

This work has been the first approach to evacuations and crowds in our research group. Future research may take advantage of the work done by this thesis and expand it with new ideas and contributions focusing on the crowd internal phenomena and performance of the simulator. Here we propose several possible future directions that can be opened.

- There is still a lack of existing data in the case of crowd turbulence. New research and data can help to improve the presented research and to calibrate the model.
- The evacuation process is rich and complex. The models can be extended and reviewed constantly including new concepts such as leaders, fire, dynamic environments, harmed people becoming obstacles, etc.
- New architectures and parallel programming models may be evaluated if they suit efficiently the model of evacuations proposed. Evaluate their performance and the advantages and disadvantages as well as their performance.
- Explore new data structures and representations of the space and their suitability for the final usage as DSS. Consider formats such as Building Information Modeling (BIM) formats and evaluate its performance, or CAD 3D models which design complex geometries. Also consider GIS localization and OpenStreet or other platforms as input of the scenario.
- Although the simulator has been validated, it would be interesting to use another methods and tools to validate with more strength the tool to increase its accuracy.
- The current application has not been used by experts. One of the main contributions will be porting the current system to experts who will use it and evaluate their needs, implement them and improve the model, the methods and the performance.
- Introduce new crowd phenomena and work towards a complete crowd simulator.

7.3 List of contributions

Here there is a list of all the papers published or accepted carried out during this research including its description and abstract. The last paper include the description of the journal paper written for this thesis which has not been finally accepted at date of submission of this thesis.

1. Albert Gutierrez-Milla, Francisco Borges, Remo Suppi, Emilio Luque, (2014) "Individual-oriented Model Crowd Evacuations Distributed Simulation", (International Conference on Computational Science, Procedia Computer Science), 29, pp. 1600-1609.

   Abstract: *Emergency plan preparation is an important problem in building design to evacuate people as fast as possible. Simulation exercises as fire drills are not a realistic situation to understand people behaviour. In the case of crowd evacuations the complexity and uncertainty of the systems increases. Computer simulation allows us to run crowd dynamics models and extract information from emergency situations. Several models solve the emergency evacuation problem. Individual oriented modelling allows to give behaviour rules to the individual and simulate interactions between them. Due to variation on the emergency situations results have to be statistically reliable. This reliability increases the computing demand. Distributed and parallel paradigms solve the performance problem. In the present work we present a crowd evacuations distributed simulator. We implemented two versions of the model. One using NetLogo and another using C with MPI. We chose a real environment to test the simulator: pavilion 2 of* Fira de Barcelona *building, able to hold thousands of persons. The distributed simulator was tested with 62,820 runs in a distributed environment with 15,000 individuals. In this work we show how the distributed simulator has a linear speedup and scales efficiently.*

2. Albert Gutierrez-Milla, Francisco Borges, Remo Suppi, Emilio Luque, (2015) "Crowd Evacuations SaaS: An ABM Approach", (International Conference on Computational Science, Procedia Computer Science), 51, pp. 473-482.

Abstract: *Crowd evacuations involve thousands of persons in closed spaces. Having knowledge about where the problematic exits will be or where the disaster may occur can be crucial in emergency planning. We implemented a simulator using Agent Based Modelling able to model the behaviour of people in evacuation situations and a workflow able to run it in the cloud. The input is just a PNG image and the output are statistical results of the simulation executed on the cloud. This allows to provide the user with a system abstraction and only a map of the scenario is needed. Many events are held in main city squares, so to test our system we chose Siena and we fit about 28,000 individuals in the centre of the square. The software has special computational requirements because the results need to be statistically reliable. Because these needs we use distributed computing. In this paper we show how the simulator scales efficiently on the cloud.*

3. Albert Gutierrez-Milla, Francisco Borges, Remo Suppi, Emilio Luque, (2015) "Crowd dynamics modeling and collision avoidance with OpenMP", (Proceedings of the 2015 Winter Simulation Conference), pp. 3128-3129.

   Abstract: *This paper deals with the problem of simulating crowd evacuations in multicore architectures. Manage evacuations is an important issue that involves lives, and in the case of crowds, thousands of lives. We present our model, able to hold thousands of agents walking through the scenario avoiding dynamic and static obstacles. We test how the model behaves with agents falling down, becoming an obstacle. In the present work we introduce a crowd model and a HPC simulation tool. We used OpenMP to program a multicore architecture.*

4. Albert Gutierrez-Milla, Francisco Borges, Remo Suppi, Emilio Luque, (2016) "Crowd Turbulence with ABM And Verlet Integration On GPU Cards", (International Conference on Computational Science, Procedia Computer Science), (accepted).

   Abstract: *Managing crowds is a key problem in a world with a growing population. Being able to predict and manage possible disasters directly affects the safety of crowd events. Current simulations focus mostly on navigation, but crowds have*

their own special characteristics and phenomena. Under specific conditions, a mass can turn into a crowd turbulence which may lead to a possible disaster. Understanding the internal phenomena is an important issue in order to model behavior. In the particular case of crowd turbulence, agents are moved by the crowd by a series of pushes, an involuntary movement that can be hard to reproduce. We propose a simple model to implement this complex problem based on intentional and involuntary interactions among the agents. The implementation is a hybrid model between the Verlet integration method and Agent Based Modeling. We implemented the proposed model using C and OpenCL and we evaluated its performance on a Nvidia GPU.

7.4 List of related publications

1. Francisco Borges, Albert Gutierrez-Milla, Remo Suppi, Emilio Luque, (2014) "Optimal Run Length for Discrete-event Distributed Cluster-based Simulations" (International Conference on Computational Science, Procedia Computer Science), 29, pp. 73-83.

   In scientific simulations the results generated usually come from a stochastic process. New solutions with the aim of improving these simulations have been proposed, but the problem is how to compare these solutions since the results are not deterministic. Consequently how to guarantee that the output results are statistically trusted. In this work we apply a statistical approach in order to define the transient and steady state in discrete event distributed simulation. We used linear regression and batch method to find the optimal simulation size. As contributions of our work we can enumerate: we have applied and adapted the simple statistical approach in order to define the optimal simulation length; we propose the approximate approach to normal distribution instead of generate replications sufficiently large; and the method can be used in other kind of non-terminating science simulations where the data either have a normal distribution or can be approximated by a normal distribution.

2. Francisco Borges, Albert Gutierrez-Milla, Remo Suppi, Emilio Luque, (2014) "A Hybrid MPI+ OpenMP Solution of the Distributed Cluster-based Fish Schooling Simulator" (International Conference on Computational Science, Procedia Computer Science), 29, pp. 2111-2120.

   Exploring the multi-core architecture is an important issue to obtaining high performance in parallel and distributed discrete-event simulations. However, the simulation features must fit on parallel programming model in order to increase the performance. In this paper we show our experience developing a hybrid MPI+OpenMP version of our parallel and distributed discrete- event individual-oriented fish schooling simulator. In the hybrid approach developed, we fit our simulation features in the following manner: the communication between the Logical Processes happens via message passing whereas the computing of the individuals by OpenMP threads. In addition, we propose a new data structure for partitioning the fish clusters which avoid the critical section in OpenMP code. As a result, the hybrid version significantly improves the total execution time for huge quantity of individuals, because it decreases both the communication and management of processes overhead, whereas it increases the utilization of cores with sharing of resources.

3. Francisco Borges, Albert Gutierrez-Milla, Remo Suppi, Emilio Luque, (2015) "Strip Partitioning for Ant Colony Parallel and Distributed Discrete-event Simulation" (International Conference on Computational Science, Procedia Computer Science), 51, pp. 483-492.

   Data partitioning is one of the main problems in parallel and distributed simulation. Distribution of data over the architecture directly influences the efficiency of the simulation. The partitioning strategy becomes a complex problem because it depends on several factors. In an Individual-oriented Model, for example, the partitioning is related to interactions between the individual and the environment. Therefore, parallel and distributed simulation should dynamically enable the interchange of the partitioning strategy in order to choose the most appropriate partitioning strategy for a specific context. In this paper, we pro-

pose a strip partitioning strategy to a spatially dependent problem in Individual-oriented Model applications. This strategy avoids sharing resources, and, as a result, it decreases communication volume among the processes. In addition, we develop an objective function that calculates the best partitioning for a specific configuration and gives the computing cost of each partition, allowing for a computing balance through a mapping policy. The results obtained are supported by statistical analysis and experimentation with an Ant Colony application. As a main contribution, we developed a solution where the partitioning strategy can be chosen dynamically and always returns the lowest total execution time.

4. Francisco Borges, Albert Gutierrez-Milla, Remo Suppi, Emilio Luque, Marylene de Brito Arduino, (2015) "An agent-based model for assessment of aedes aegypti pupal productivity" (Proceedings of the 2015 Winter Simulation Conference), pp. 159-170.

   The dengue is a febrile disease whose main vector transmitter is the Aedes Aegypti mosquito. This disease has an annual register of 50 million infections worldwide. Simulations are an important tool in helping to combat and prevent the epidemic and, consequently, save lives and resources. Therefore, in this paper, we propose an agent-based model for assessment of the pupal productivity of the Aedes Aegypti mosquito. In this model, the reproduction of the mosquito takes into account the productivity of each type of container. The preliminary results show the effects of considering the pupal productivity for the control and prevention of dengue. As a result, we observed that the prevention methods must consider pupal productivity and that the distance between containers might leverage productivity and increase transmission risk. We verify the completeness and functionality of the model through experimentation using NetLogo.

References

[1] WikiMedia Loveparade 2010. https://commons.wikimedia.org/wiki/File:2010_07_24_arne_mueseler_0223.jpg. licensed under Creative Commons by WikiMedia.

[2] Benigno E Aguirre, Manuel R Torres, Kimberly B Gill, and H Lawrence Hotchkiss. Normative collective behavior in the station building fire. *Social science quarterly*, 92(1):100–118, 2011.

[3] Hani Alnabulsi and John Drury. Social identification moderates the effect of crowd density on safety at the hajj. *Proceedings of the National Academy of Sciences*, 111(25):9091–9096, 2014.

[4] T. Meyer-König B. Mandt. Evacuation analysis. theatre in tampere finnland. TraffGo GmbH, 2002.

[5] Avi Bleiweiss. Multi agent navigation on the gpu. In *GDC09 Game Developers Conference*, volume 2009, 2009.

[6] Maik Boltes, Mohcine Chraibi, Stefan Holl, Armel Ulrich Kemloh Wagoum, Gregor Lämmel, Weichen Liao, Wolfgang Mehner, Antoine Tordeux, and Jun Zhang. Experimentation, data collection, modeling and simulation of pedestrian dynamics.

[7] Albert Hadley CANTRIL, Hazel Gaudet, Herta Herzog, and Howard Koch. *The Invasion from Mars. A Study in the Psychology of Panic. With the Complete Script of the Famous Orson Welles Broadcast (Howard Koch's Freely Adapted Version of HG Well's War of the Worlds). By H. Cantril, with the Assistance of Hazel Gaudet & Herta Herzog.* Princeton University Press, 1940.

[8] Chris Cocking and John Drury. Talking about hillsborough:'panic'as discourse in survivors' accounts of the 1989 football stadium disaster. *Journal of Community & Applied Social Psychology*, 24(2):86–99, 2014.

[9] coolwojtek. Love parade teil 5 2010. https://www.youtube.com/watch?v=xxd_KlaCiNY, 2010.

[10] Tom Cox, Jonathan Houdmont, and Amanda Griffiths. Rail passenger crowding, stress, health and safety in britain. *Transportation Research Part A: Policy and Practice*, 40(3):244–258, 2006.

[11] El Baúl de Josete. https://elbauldejosete.files.wordpress.com/2008/06/200705012243322.jpg. licensed under Creative Commons by El Baúl de Josete.

[12] Ignacio Díaz-Emparanza. Selección del número de replicaciones en un estudio de simulación. *Estadística española*, 37(140):497–509, 1995.

[13] JF Dickie. Major crowd catastrophes. *Safety science*, 18(4):309–320, 1995.

[14] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 29(5):485–501, 2010.

[15] John Drury, Chris Cocking, and Steve Reicher. Everyone for themselves? a comparative study of crowd solidarity among emergency survivors. *British Journal of Social Psychology*, 48(3):487–506, 2009.

[16] Russell R Dynes. Panic and the vision of collective incompetence. *Natural Hazards Observer*, 31(2):5–6, 2006.

[17] Thunderhead Engineering. http://www.thunderheadeng.com/pathfinder/, 2016.

[18] Ugo Erra, Rosario De Chiara, Vittorio Scarano, and Maurizio Tatafiore. Massive simulation using gpu of a distributed behavioral model of a flock with obstacle avoidance. In *Proceedings of vision, modeling and visualization*, volume 2004, 2004.

[19] Legion Evac. http://www.legion.com/legion-evac. 2016.

[20] EXODUS. http://fseg.gre.ac.uk/exodus/, 2016.

[21] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17 (7):760–772, 1998.

[22] Fira de Barcelona authors. Fira de barcelona. https://www.firabarcelona.com/, last viewed January 2014.

[23] JJ Fruin. The causes and prevention of crowd disasters.. 2002. online, 2002.

[24] Vincent A Fusaro, Prasad Patil, Erik Gafni, Dennis P Wall, and Peter J Tonellato. Biomedical cloud computing with amazon web services. *PLoS Comput Biol*, 7(8):e1002147, 2011.

[25] Abhinav Golas, Rahul Narain, and Ming C Lin. Continuum modeling of crowd turbulence. *Physical Review E*, 90(4):042816, 2014.

[26] Stephen J Guy, Jatin Chhugani, Changkyu Kim, Nadathur Satish, Ming Lin, Dinesh Manocha, and Pradeep Dubey. Clearpath: highly parallel collision avoidance for multi-agent simulation. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 177–187. ACM, 2009.

[27] Health, Safety Commission, et al. The hillsborough incident-collapse load calculations for barrier 124a. 1989.

[28] Dirk Helbing and Pratik Mukerji. Crowd disasters as systemic failures: analysis of the love parade disaster. *EPJ Data Science*, 1(1):1–40, 2012.

[29] Dirk Helbing, Illés Farkas, and Tamas Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, 2000.

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

[31] Dirk Helbing, Anders Johansson, and Habib Zein Al-Abideen. Dynamics of crowd disasters: An empirical study. *Physical review E*, 75(4):046109, 2007.

[32] LF Henderson. The statistics of crowd fluids. *Nature*, 229:381–383, 1971.

[33] Colin M Henein and Tony White. Macroscopic effects of microscopic forces between agents in crowd models. *Physica A: statistical mechanics and its applications*, 373:694–712, 2007.

[34] RL Hughes. The flow of large crowds of pedestrians. *Mathematics and Computers in Simulation*, 53(4):367–370, 2000.

[35] Roger L Hughes. A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological*, 36(6):507–535, 2002.

[36] Crowd Dynamics Myriad II. http://www.crowddynamics.com/spatial-analysis-module.php, 2016.

[37] Binu Jacob, Anthony R Mawson, Marinelle Payton, and John C Guignard. Disaster mythology and fact: Hurricane katrina and social attachment. *Public health reports (Washington, DC: 1974)*, 123(5):555–566, 2007.

[38] Thomas Jakobsen. Advanced character physics. In *Game Developers Conference*, pages 383–401, 2001.

[39] Norris R Johnson. Panic at" the who concert stampede": An empirical assessment. *Social Problems*, pages 362–373, 1987.

[40] Fred Jopp, Sven Erik Jørgensen, Melanie Trexler, Hauke Reuter, Donald DeAngelis, and Broder Breckling. *Modelling complex ecological dynamics: an introduction into ecological modelling for students, teachers & scientists*. Springer Science & Business Media, 2011.

[41] JuPedSim. http://www.jupedsim.org, 2016.

[42] Sujeong Kim, Stephen J Guy, Wenxi Liu, Rynson WH Lau, Ming C Lin, and Dinesh Manocha. Predicting pedestrian trajectories using velocity-space reasoning. In *Algorithmic Foundations of Robotics X*, pages 609–623. Springer, 2013.

[43] Mariam Kiran, Kabiru Maiyama, Haroon Mir, Bashir Mohammad, and AA Oun. Agent-based modelling as a service on amazon ec2: Opportunities and challenges. In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pages 251–255. IEEE, 2015.

[44] Tobias Kretz, Cornelia Bönisch, and Peter Vortisch. Comparison of various methods for the calculation of the distance potential field. In *Pedestrian and evacuation dynamics 2008*, pages 335–346. Springer, 2010.

[45] Erica D Kuligowski, Richard D Peacock, and Bryan L Hoskins. *A review of building evacuation models*. US Department of Commerce, National Institute of Standards and Technology Gaithersburg, MD, 2005.

[46] Paul V Lemkau. Epidemiology of hysteria. In *Psychiatric Forum*, volume 3, pages 1–14. WILLIAM S HALL PSYCHIAT INST S CAROLINA DEPT MENTAL HEALTH BOX 119, COLUMBIA, SC 29202, 1973.

[47] Robert Lubaś, Marcin Mycek, Jakub Porzycki, and Jarosław Wąs. Verification and validation of evacuation models–methodology expansion proposition. *Transportation Research Procedia*, 2:715–723, 2014.

[48] Peter Lynch et al. Richardson's forecast factory: the $64 000 question. *Meteorological Magazine*, 122:69–69, 1993.

[49] Salon Manga Barcelona XIX mapa palacio. http://manga-xix.ficomic.com/.

[50] Anthony R Mawson. Understanding mass panic and other collective responses to threat and disaster. *Psychiatry*, 68(2):95–113, 2005.

[51] MIT. Star - software tools for academics and researchers: Cluster. http://star.mit.edu/cluster/, 2010.

[52] Betty J Mohler, William B Thompson, Sarah H Creem-Regehr, Herbert L Pick Jr, and William H Warren Jr. Visual flow influences gait transition speed and preferred walking speed. *Experimental brain research*, 181(2):221–228, 2007.

[53] John Morris. https://www.flickr.com/photos/jm999uk/2872998856. licensed under Creative Commons by John Morris.

[54] Carol E Nicholson and B Roebuck. The investigation of the hillsborough disaster by the health and safety executive. *Safety Science*, 18(4):249–259, 1995.

[55] Thomas W. Paradis. *Living the Palio: A Story of Community and Public Life in Siena, Italy*. iUniverse, 2014.

[56] Nuria Pelechano, Kevin O'Brien, Barry Silverman, and Norman Badler. Crowd simulation incorporating agent psychological models, roles and communication. Technical report, DTIC Document, 2005.

[57] Nuria Pelechano, Jan M Allbeck, and Norman I Badler. Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 99–108. Eurographics Association, 2007.

[58] Michael J Quinn, Ronald A Metoyer, and Katharine Hunter-Zaworski. Parallel implementation of the social forces model. In *Proceedings of the Second International Conference in Pedestrian and Evacuation Dynamics*, pages 63–74. Citeseer, 2003.

[59] Bruce Cameron Reed. *the Physics of the Manhattan Project*. Springer, 2014.

[60] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH computer graphics*, volume 21, pages 25–34. ACM, 1987.

[61] Paul Richmond, Simon Coakley, and Daniela M Romano. A high performance agent based modelling framework on graphics card hardware with cuda. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 1125–1126. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[62] Karl Erik Rosengren, Peter Arvidson, and Dahn Sturesson. The barsebäck'panic': A radio programme as a negative summary event. *Acta Sociologica*, 18(4):303–321, 1975.

[63] Salo del manga authors. Salo del manga. http://manga-xix.ficomic.com/, last viewed January 2014, 2013.

[64] Integrated Environmental Solutions SIMULEX. https://www.iesve.com/software/ve-for-engineers/module/Simulex/480, 2016.

[65] Jamie Snape, Jur P van den Berg, Stephen J Guy, and Dinesh Manocha. Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles. In *IROS*, pages 5917–5922, 2009.

[66] Jamie Snape, Stephen J Guy, Ming C Lin, and Dinesh Manocha. Local and global planning for collision-free navigation in video games. In *Planning in Games Workshop. Citeseer*, pages 7–10. Citeseer, 2013.

[67] Oassys MassMotion software. http://www.oasys-software.com/products/engineering/massmotion.html, . 2016.

[68] STEPS software. http://www.steps.mottmac.com/, . 2016.

[69] WikiMedia Mahamaham stampede. https://commons.wikimedia.org/wiki/File:Mahamaham_Festival_in_Kumbakonam.jpg. licensed under Creative Commons by WikiMedia.

[70] Pablo Cristian Tissera, A Marcela Printista, and Emilio Luque. A hybrid simulation model to test behaviour designs in an emergency evacuation. *Procedia Computer Science*, 9:266–275, 2012.

[71] TOP500.org. http://top500.org/, 2016.

[72] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. In *ACM Transactions on Graphics (TOG)*, volume 25.3, pages 1160–1168. ACM, 2006.

[73] http://www.aboutsiena.com/palio-of-Siena.html. The palio of siena, 2015.

[74] http://www.teatterikesa.fi/. Tampere theater, 2015.

[75] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1928–1935. IEEE, 2008.

[76] Isabella von Sivers, Anne Templeton, Gerta Köster, John Drury, and Andrew Philippides. Humans do not always act selfishly: social identity and helping in emergency evacuation simulation. *Transportation Research Procedia*, 2: 585–593, 2014.

[77] Armel Ulrich Kemloh Wagoum, Mohcine Chraibi, Jonas Mehlich, Armin Seyfried, and Andreas Schadschneider. Efficient and validated simulation of crowds for an evacuation assistant. *Computer Animation and Virtual Worlds*, 23(1):3–15, 2012.

[78] WikiMedia The way to Jamarat Bridge. https://en.wikipedia.org/wiki/File:The_way_to_Jamarat_Bridge_3.JPG. licensed under Creative Commons by WikiMedia.

[79] Henry Weckman, Simo Lehtimäki, and Seppo Männikkö. Evacuation of a theatre: Exercise vs calculations. *Fire and Materials*, 23(6):357–361, 1999.

[80] Uri Wilensky. Netlogo. 1999.

[81] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10, 2013.

[82] Wenjian Yu and Anders Johansson. Modeling crowd turbulence by many-particle simulations. *Physical review E*, 76(4):046105, 2007.