博士論文

Enhancements and applications of a scalable multi-agent based large urban area evacuation simulator with emphasis on the use of cars

(スケーラブルなマルチエージェント大都市域避難行動シミュレータの自動車の考慮に重点をおいた拡張と適用)

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF CIVIL ENGINEERING
OF THE GRADUATE SCHOOL OF ENGINEERING,
UNIVERSITY OF TOKYO
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Aguilar Melgar Leonel Enrique
アギラール・メルガール　レオネル　エンリケ
March 2015

ENHANCEMENTS AND APPLICATIONS OF A SCALABLE MULTI-AGENT BASED LARGE URBAN AREA EVACUATION SIMULATOR WITH EMPHASIS ON THE USE OF CARS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF CIVIL ENGINEERING
OF THE GRADUATE SCHOOL OF ENGINEERING,
UNIVERSITY OF TOKYO
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Signature	Date	Seal
Advisor:		
Co-Advisor:		

Leonel Enrique Aguilar Melgar
March 2015

Acknowledgement

First and foremost, I would like to express my sincere and deep gratitude to Dr. Lalith Wijerathne and Dr. Muneo Hori, whose expertise and passion in academia brought constant intellectual and critical discussion to my research. This provided a thought provoking environment that allowed this dissertation to come into existence.

> "I much prefer the sharpest criticism of a single intelligent man to the thoughtless approval of the masses." - Johannes Kepler

Moreover, I would like to thank my supervisor Dr. Lalith Wijerathne for his relentless work, a truly mentoring example worth of my deepest admiration. His outstanding knowledge and his invaluable guidance made this dissertation possible.

I would also like to thank Dr. Tsuyoshi Ichimura, Dr. Seizo Tanaka and Dr. Hiromichi Nagao for their guidance, teaching and help throughout my studies. I would like to express my sincere gratitude to my doctorate committee members Dr. Masahide Horita, Dr. Takashi Oguchi, Dr. Lalith Wijerathne, Dr. Muneo Hori, Dr. Tsuyoshi Ichimura and Dr. Daichi Yanagisawa whose brilliant comments, suggestions and criticism provided guidance and challenges to this research.

I would like to thank Japan's MEXT scholarship for providing me with the necessary funding to accomplish this research; and the faculty of Del Valle University, Guatemala, Maria Eugenia de Nieves and Nancy de Zurita head of the mathematics department and Roberto Godo head of the civil engineering department for their support in obtaining the funding.

I would like to thank my fellow lab members and friends whose support enabled me to keep my research moving forward. I extend my gratitude especially to Ryoichiro Agata, Kohei Fujita, Stephen Jacob, Supun Chamara and Mahendra Pal.

Special thanks to my personal heroes my father Leonel Aguilar and my mother Gloria Melgar; their example, sacrifice and support allowed me to reach this point. My sincere gratitude is also extended to my family, especially to, Nereida Aguilar, Jorge Diaz, Rosa Elena Solis for their support and encouragement throughout all these years.

Finally, I would like to express my gratitude to RIKEN's Advanced Institute for Computational Science[1], the Earthquake Research Institute at the University of Tokyo[2], and The

University of Tokyo[3] for providing access to the high performance computing infrastructure necessary for the development and testing of this work. (K computer[1], EIC[2], Oakleaf[3], Oakbridge[3]).

Abstract

This work presents the enhancements of a scalable multi-agent based large urban area evacuation simulator, with the aim of quantitatively evaluating mixed mode evacuations with cars and pedestrians. This dissertation introduces a mathematical framework to provide a language and notation necessary to explain the multi-agent system and the enhancements required to evaluate mixed mode evacuations; provides an overview on the design of agents and environment and the enhancements; introduces the concept of basic translational interactions and provides means to automatically calibrate them based on field observations; and finally demonstrates the ability to perform quantitative evaluations of mixed mode evacuations evaluating different evacuation strategies and countermeasures. The design and implementation of the multi-agent system, verification and validation of the necessary components, and demonstrative applications are presented.

During the 2011 Great East Japan Earthquake and Tsunami, many residents living in coastal areas used cars to evacuate, reportedly producing congestion. This experience has drawn the attention to the use of cars for tsunami triggered evacuations with the aim to identify means to benefit from the use of cars, especially to evacuate people in need. There have been discussions on the implementation of countermeasures and evacuation strategies in mixed mode evacuations, but currently there is no way to make detailed quantitative measures of the outcome of such measures.

Existing studies on the use of cars are limited to traffic flow studies in highways and main roads, where vehicles have no or little interaction with pedestrians. No research articles are found on the use of cars in narrow roads of densely populated residential areas, where the interaction among pedestrians and vehicles can slow each other down producing significant impacts on the evacuation throughput. The lack of research on this issue can be attributed to the limitations in common evacuation simulation software such as the use of simplified models for large areas (one dimensional networks or queues) or the software considering details in the simulation not being able to scale to large urban areas.

In order to address these limitations a multi-agent system based mass evacuation simulation software is enhanced to include the detailed interactions among evacuees. This simulator can accommodate sub-meter detailed model of environment and utilizes high performance computing to simulate large urban areas. Different types of agents with various interaction models are introduced with the aim to model the heterogeneity of crowds and the different modes of evacuation.

For the basic interaction, the avoidance of collisions, a collision avoidance scheme, ORCA, is enhanced and adapted to reproduce field observations. For the special case of car-pedestrian interactions, field observations are performed in order to collect quantitative data necessary for validating the implemented interaction model. Pedestrian-pedestrian interaction is validated by comparing the simulation results to the field observations by Mori and Tsukaguchi, and Weidmann's summary of field observations. Car-car interaction is validated by comparing the simulation results with Lincoln tunnel observations, and the car-pedestrian interactions are validated by comparing the simulation results with observations collected by the author.

Evacuees' collision avoidance behaviors change according to many factors like culture, age, time of the day, etc., and can be reproduced by adjusting several parameters in the implemented algorithms. A calibration tool, enhanced with parallel computing capabilities, is implemented to automate the identification of these parameters according to given observation data.

In order to support the agents' interaction models, the use of vertical shelters and multi-storey buildings, an extended hybrid environment model consisting of a high resolution grid and topological graphs is introduced. Also, tools are implemented to automate the construction of the hybrid environment from GIS data.

Finally, demonstrative applications of the software highlighting the need of detailed modeling and its potential in finding means to benefit from the usage of cars in mixed mode evacuations are presented. Strategies and countermeasures such as the restriction of the car usage for people in need, widening of roads and the use of different evacuation schedules are evaluated and quantified. It is found that the best throughput after 40 minutes of evacuation is provided by restricting the usage of cars to people in need with a vehicle usage of 25% percent of the population. Moreover, the initial throughput of the evacuation can be improved by persuading people evacuating using vehicles to start the evacuation earlier, providing a boost on the initial throughput of more than 10\% within the first 20min of the evacuation. Additionally the effect of the car-pedestrians interactions and the effect of the model of intersections are quantified amounting to about 5% and 4% respectively for the base demonstration scenario. Furthermore, the number of Monte-Carlo simulations necessary to provide robustness to the results in this scenario are evaluated. It is found that for the base scenario Monte-Carlo simulations drawing 600 samples are necessary to provide a converged characterization of the distribution of the results.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Extensive studies of a wide range of possible evacuation scenarios and preparations are necessary to prevent large loss of lives from the anticipated mega-tsunamis. In Japan, although “tsunami tendenko” (save yourself ignore the others) has been instilled and the use of cars for evacuation is forbidden, during 2011 Great East Japan Earthquake and Tsunami many people evacuated using cars in an attempt to save their families, their assets or because of their perceived inability to reach an evacuation area in time. This caused unexpected problems during the evacuation, highlighting the need to consider more realistic scenarios including every detail influencing the evacuation time significantly. Current software for large area evacuation simulations use highly simplified models, in which environment and evacuees are modeled as 1D networks and queues. Although these simplified models give useful insights for certain scenarios, they fail to capture the complex interactions arising in narrow urban streets between evacuees using different modes of evacuation. Moreover, software capable of taking details into account are restricted to smaller domains as they don’t scale beyond a few CPU cores[2]. There is a great need for a scalable large urban area simulator that both supports detailed models of environment and evacuees and represents the different modes of evacuation as observed in reality.

In order to address this need an evacuation simulation software is enhanced to provide the functionality to evaluate mixed mode evacuations including the different interactions between the evacuees and the environment. These enhancements have aim to provide expert teams with a software usable as an experimental playground to quantitatively evaluate the effects of different scenarios and mitigation measures in an exploratory manner. For this purpose, an existing evacuation simulation software, part of the integrated earthquake simulator (IES)[5] , is used as the base framework. This software is selected due to its characteristics and performance, namely: its multi-agent system paradigm base, which allows the introduction of heterogeneity in crowds ideal for the introduction of complex agents representing vehicles and the ability to extend the existing pedestrians agents to interact with each other and the environment; the ability to include sub-meter in large urban environments, which necessary to support the modeling of detailed evacuees' behaviors and interactions; and the provision of a scalable parallel computing framework, necessary to handle the work load of performing Monte Carlo simulations of scenarios including hundreds of thousands of agents with complex interactions in a large urban environment.

In addition to introducing evacuation using cars the basic agents' interactions which have a significant effects on the evacuation time have been included, e.g. collision avoidance[3], vision[3], navigation[3] and communication. Other enhancements include incorporating agents' past experiences in their decision making [41], inclusion of dynamic changes of environment such as tsunami inundation[41], visual identification of road blockages [41], etc. Additionally, an HPC enhanced tool to automatically calibrate the model to the target population, application scenario, etc., are developed. In order to provide reliability to the software, implemented interactions are calibrated and validated by comparing them to observations[3]. Furthermore, a scalable parallel computing extension is developed[6]. Furthermore to account for the uncertainties and introduce robustness Monte-Carlo simulation capability is implemented .

The rest of the dissertation is organized as follows: Chapter 2 provides a short description of the method used as the framework and the mathematical framework extended in this work to describe it. Chapter 3 introduces the enhancements provided to the software starting with formalizing the design of the enhancements to the software and the implemented tools created to support the design of behavioral and environmental scenarios. Chapter 4, gives further details on the development and extension of the collision avoidance scheme used as the base for certain agent's interactions as well as the derived interactions and their validation by comparing them to field observations. This chapter also presents the automatic tuning through the development and use of an HPC enhanced automatic calibration. Chapter 6 presents demonstrative applications of the evacuation simulation software to highlight the need of detailed modeling and simulation. Chapter 7 provides additional remarks and concludes this dissertation.

Chapter 2

The evacuation simulation framework

2.1 Multi-agent system

There are many competing paradigms widely used in the simulation of crowds and other social phenomena, each of them varying fundamentally in the way they model the underlying phenomena. From those the most relevant are: differential equations, cellular automata, graph networks and queues, agent based models, and multi-agent systems. Differential equations, are widely used to model crowds. They are based on the assumption that crowd movements can be modeled as a fluid, disregarding evacuees' individualities. Graph networks and queues follow a similar principle modeling the evacuation as a flow of evacuees via nodes and edges controlled by certain set of rules. These models are usually enhanced by hybridizing them and using autonomous agents that choose which queue to enter. Cellular automata restricts the domain to cells and simulate evacuees as changes in the states of the cells. Traditional cellular automata usually rely on the same transition function or rules to model every evacuee, which disables the ability to model heterogeneity. Even under the current trend of heterogenous cellular automata these models are highly restricted to the domain and the definition of the cells. Although all these models provide useful information about the evacuation process, the underlying simplifications restrict their applications.

Multi-agent based evacuation simulation takes advantage of the ability of this method to provide heterogeneity in crowds by providing every agent with different functionalities and instantiating every agent with unique properties if required. This ability of considering each individual's autonomous decisions and actions allows the exploration of the impact of various behavioral or interaction models in the simulation. Furthermore, it allows exploration of a wide range of complex scenarios, by introducing a detailed model of the environment and related agents functionalities to interact with the environment and each other. The access to fine grained information of the simulation makes easier to spot incongruences in evacuees or environment models.

This versatility of multi-agent systems comes with a number of disadvantages, such as the difficulty to develop the system and design algorithms, the difficulty to verify and validate such models, etc. Although this difficulty could be also present in the other paradigms it is specially aggravated when introducing detailed modeling in multi-agent systems due to its complexity. Furthermore, it is generally accepted that in the simulation of a social phenomena, it is difficult/impossible to assure that the actions occurring in the simulation match reality. Therefore, assumptions over the autonomous actions and behaviors are inexorable. These assumptions are usually gross or completely avoided in the case of the simplified models as their perspective is macroscopic. Additionally for large scale simulations it is impossible to foresee every possible combination of interaction, states combination and situations which requires constant improvement of the system to handle these special cases. Moreover the inclusion of details in the environment and agents able to process, analyze and use this data incurs into a high computational cost. All these disadvantages are handled or mitigated with different techniques and tools, for example:

- the use of HPC to handle the high computational cost,
- the use of a hybrid environment to be able to represent details and provide efficiency in its usage,
- the introduction of a mathematical framework to provide a notation and means to specify, compare, and separate every component of the system, necessary for the specification, comparison and validation of the individual components
- throughout validation and calibration of the interaction models and emergent phenomena to provide results as close to observations as possible, among others.

2.1.1 Terminology

There are still ongoing philosophical and technical debates on what constitutes a Multi Agent System, an Agent Based Model, an Intelligent System, Interactions and even an Agent itself. These terms are loosely used in a range of works. For example, [23] defines a Multiagent system as a distributed computing system, composed of "intelligent" "interacting" entities, and goes further into defining "intelligence" based on the interacting entities being able to take intentional stances or to posses a distinct knowledge level. [24] , acknowledges the existence of competing mutually inconsistent definitions and provides a loose definition:

> "Multiagent systems are those systems that include multiple autonomous entities with either diverging information or diverging interests, or both"

In the case of the software used as a base for this work its main constituent are autonomous agents confined in the same domain subjected to different types of interactions among them. This collection of interacting intelligent autonomous agents and environment will be referred as the multi-agent system (MAS) in this work.

2.2 Model

The basic elements of the model are the agents and the environment that surrounds them. The agents represent evacuees escaping to shelters either on foot or vehicles, volunteers or officials assisting the evacuees, etc. Each agent has different abilities, responsibilities and roles. The agents gather information from their environment, and interact both with it and other agents, while trying to attain their goals, e.g. reach a safe destination, encourage other evacuees to start evacuating, guide other groups of evacuees.

The code is developed using C++, since the object oriented features are versatile in implementing a system with heterogeneous and complex agents. The agents and their components are encapsulated in specialized objects.

2.2.1 Environment

The environment is modeled with a high resolution two dimensional grid; presently $1m \times 1m$ grid spacing is used. The grid cells hold one of the three different states: occupied, unoccupied or exit. Highly localized dynamic changes such as the reduction of the road width due to collapsed buildings, fire, etc. can be easily modeled in a grid environment. At the same time, it allows modeling of complex agent interactions and the use of all the available space. Further, the grid can be easily distributed among processes in a parallel computing environment.

2.2.2 Agents

The agents are the software counterpart of the evacuees. All the agents share three basic functionalities: *see*, *think* and *act*. See, identifies the boundary of visibility and the agents in the visible neighborhood; think, evaluates the available options based on the information collected in *see* and decides the action to be taken based on that; and act updates an agent's state according to the actions chosen by think.

The agents can be categorized into two main groups based on their previous knowledge of the environment: visitors, who do not have any prior information of the environment and non-visitors, such as residents and officials, who are familiar with the environment. These two agent groups provide a crude example for the implementation of specialized agent types, and serve as a base for the agents' enhancements.

All the agents contain a set of classes representing ability, memory, recognition, vision and navigation. Further, these classes are specialized depending on the agent type, responsibilities, age, etc. of an agent. The ability object contains information that pertains to inherent properties of the agent such as maximum speed and their maximum sight distance. The memory object holds agent's past experiences and present cognitive state variables, such as their current direction, their desired final destination and, in the case of non-visitor agents, their path to the final destination. The recognition object holds information of perceived surroundings, information about the neighbors and exits in their visible surrounding. The vision object contains information and functions to process the information regarding their environment. Finally, the navigation object provides the required functions to navigate in the grid environment.

2.3 Mathematical Framework

The multi-agent system is time step driven and architecture wise the agents would fall in the category of cognitive agents. The implementation of cognition and behavioral models are not part of the software itself, but considered as exploratory scenarios. This section presents the mathematical framework of the system, based upon the dynamical system framework by Laubenbacher et. al. [4].

Let $A = \{a_1, a_2, ..., a_n\}$ represent the set of all the n agents in a simulation. Considering i^{th} agent's state composed of internal (personal or inaccessible to other agents) and external (observable or deducible by other agents) states, $s_i = \{s_i^{int}, s_i^{ext}\}$; defining its neighborhood as $N_i = \{N_i^{env}, N_i^{agent}\}$, consisting of environment and agent neighborhood, which are defined as the region of the environment and the agents it can interact with, respectively; and denoting $x_i = \{s_i, N_i\}$ as the agent's local system state in the local system space, $x_i \in X_i^*$; the agent execution can be conceptualized as:

$$f_i : X_i^* \to X_i^*, \tag{2.1}$$

$$Env^* = \bigcup_{i=1}^{n} N_i^{env} \tag{2.2}$$

where the time evolution of the agent's local system is given by $x_i^{t+\Delta t} = f_i(x_i^t)$. The system's active environment is then described by $Env^* = \bigcup_{i=1}^{n} N_i^{env}$. Although formally the active environment is a subset of the whole environment, $Env^* \subseteq Env$, for practical purposes they will be referred indistinctly. The evolution of dynamical environment is defined by a set of update functions. Dynamical changes of the environment are introduced with several update functions λ_i's which perform the time evolution mapping

$$\lambda_j : Env \to Env. \tag{2.3}$$

The state space of the whole system is the assembly of all local system state spaces, $X^n = \bigcup_{i=1}^{n} X_i^*$. By defining the evolution function of the system state as the assembly of all local and environment update functions, the following discrete dynamical system is built.

$$\phi = (f_1, f_2, ..., f_n, \lambda_1, \lambda_2, ..., \lambda_m) : X^n \to X^n \tag{2.4}$$

Agents' dependency graph, G_{dep}, is formed by considering agents as vertices and forming links between any pair of interacting agents based on proximity, visibility, communication, etc.

$$G_{dep} = (A, E) \tag{2.5}$$

where,

$$E = \{(a,b) | (a \neq b) \wedge (a,b \in A) \wedge ((N_a^{env} \cap N_b^{env} \neq \emptyset) \vee ((a \in N_b^{agent}) \vee (b \in N_a^{agent})))\}$$

In this context, an agent has been formalized as $a = \{f, s\}$, the a_i's local system as $syst_{local} = \{f_i, x_i\}$, and the multi-agent system $syst = \{\phi, x^n\}$ where $x^n \in X^n$. If interfering concurrent interactions take place, additional specifications on the assembly of the global state have to be provided.

2.3.1 In the context of tsunami evacuation simulation

As exposed in [4], every permutation in the order of application of f_i defines a different dynamical system. In many cases, such as tsunami evacuation simulation, the order dependence in the application of f_i would signify the inclusion of unacceptable unfairness in the simulation. Hence, the system is formulated as a parallel discrete dynamical system.

Heterogeneity in crowds is introduced by varying relevant components of each agent's state, s, and/or the local update functions, f. As it would be impractical to specialize every f_i and s_i non intersecting subsets, $F^\tau \subseteq F$, $S^\tau \subseteq S$ are created, where $F = \{f_1, .., f_n\}$ and $S = \{s_1, ..., s_n\}$. τ stands for the agent type label. From now on $a^\tau = \{f^\tau, s^\tau\}$ will refer to a representative member of the specialized subgroup with label τ. f^τ is specialized based on the role of an agent and the information it posses, while s^τ is specialized based on agent's physical capabilities and role. Furthermore, members of each subgroup may have different components of s_i, like maximum speed, age, etc., making even the subgroup members to be heterogeneous.

The elements of the dynamical system are then crafted to model different scenarios, which are of interest to be investigated. A base scenario consists of the essential elements to be explored such as environment damage models, mitigation measures, essential initial conditions, and the set of local update functions encompassing different behavioral models. The unessential elements are then used to represent the stochastic nature of the phenomenon of study. In other words, although a specific population density might be essential for a given scenario, the actual position age, etc. might be considered unessential, thus used to introduce randomness to the model. In this formalism, the evacuation simulation system can be abstracted as a configuration reachability problem, in which, given a scenario, it is analyzed if undesired states (bottlenecks, high percentage of casualties, etc.) or desired states (high percentage of agents evacuated, feasible retrofitting costs, etc.) have been reached.

The exploration of the complete phase space is impossible as there are an uncountable number of initial conditions and possible states. Therefore, the analysis is constrained to the most probable trajectories inside this dynamical system. Monte Carlo simulations are performed, leaving the essential parameters of the scenario unchanged in order to analyze the stability of these trajectories and their end states and provide statistical grounds to the robustness of the results. It is of special interest to quantify the stability of the states of interest, as stable states provide robustness in the decision making (while keeping in mind the limitations of the underlying model).

2.3.2 The general agent's state - s

Each agent possess a unique state s which is composed of an internal state s^{int} and an external state s^{ext}. s^{int} encompasses an agent's desired moving direction, desired speed, past experiences such as blocked roads, recognized neighbors, what it has identified in it's visible neighborhood, etc. s^{int} is inaccessible to other agents. The external state s^{ext} is the information which other agents can observe, such as position, physical extension, current moving direction and other features that are deducible by the neighbors such as speed. The physical occupancy of pedestrians are modeled as circles.

2.3.3 The general local update function - f

the implemented agent based model stems from the local system update functions f. An agent's local system update function encompasses its behavior, actions and interactions models, aggregated in *See*, *Think* and *Act*. f is composed of a set of more basic functions g^i's, referred as constituent functions; $f = g^1 \circ g^2 \circ ... \circ g^m$. Some of the presently implemented constituent functions are briefly explained below. Further explanation of some of these constituent functions can be found in section 5.

g^{eyes}**:** Scans N_i^{env} and creates a visual boundary in s_i^{int} based on i^{th} agent's sight distance, which is 50m or longer.

$g^{identify\_env}$**:** Scans visual boundary and extracts features such as open paths.

$g^{navigate}$**:** Chooses which path to take out of those identified in $g^{identify\_env}$, based on an agent's destination.

$g^{identify\_inter}$**:** Recognizes agents to interact with, based on visibility, interaction radius, etc.

g^{coll_av}**:** Finds a collision free velocity to move along the path chosen in $g^{navigate}$, evading collision with neighbors which are identified in $g^{identify\_inter}$.

$g^{path\_planning}$**:** Finds paths with satisfying different needs such as the shortest path, perceived to be safest, etc. , see section 5.2.2.3.

$g^{is\_path\_blocked}$**:** Visually identifies whether desired path is blocked, see section 5.2.2.3.

$g^{find\_followee}$**:** Finds an agent to follow, see section 5.2.2.3.

$g^{find\_and\_follow}$**:** Finds a suitable agent and follows it, see section 5.2.2.3.

$g^{deliver\_message}$**:** Delivers a message to another agent, the message can contain an order to evacuate or further information about the environment such as blocked paths, etc.

$g^{seek\_not\_evacuating}$**:** Seeks for agents who have not started to evacuate, used by agents representing figures of authority in order to trigger the evacuation of other agents using, $g^{deliver\_message}$.

$g^{execute\_actions}$**:** Executes desired actions such as move

g^{update}**:** Updates an agent's state

See consists of g^{eyes}. *Think* encompasses the exploratory behavioral scenarios providing a basic workbench for an agent designer to focus on the cognition and behavioral models of the agents by applying any of the predefined constituent functions or adding new. The basic constituents of *Think* are $g^{identify\_env}$, $g^{navigate}$, $g^{identify\_inter}$ and g^{coll_av}. *Act* is composed of $g^{execute\_actions}$ and g^{update}. By specializing *Think*, from now on referred as behavioral model, different agent types, τ's, are introduced. An example is presented in Figure 2.1.

2.3.4 Pseudo agent execution parallelism

As mentioned in 2.3.1 there is a need to execute the local system update functions f_i in parallel. Obviously this is possible only in theory as in practice any scalable implementation cannot execute every agent in parallel. This restriction is due to every f_i requiring different execution time, the hardware requirements being impractical or not scalable, etc. In order to mimic the parallelism required the agent execution is separated in two phases, the execution phase, and the update phase. In order to achieve this pseudo parallelism components of s_i affected by the execution of f_i are duplicated and a copy is kept as current, and next state (after the application of the update function). This duplication of states is of special

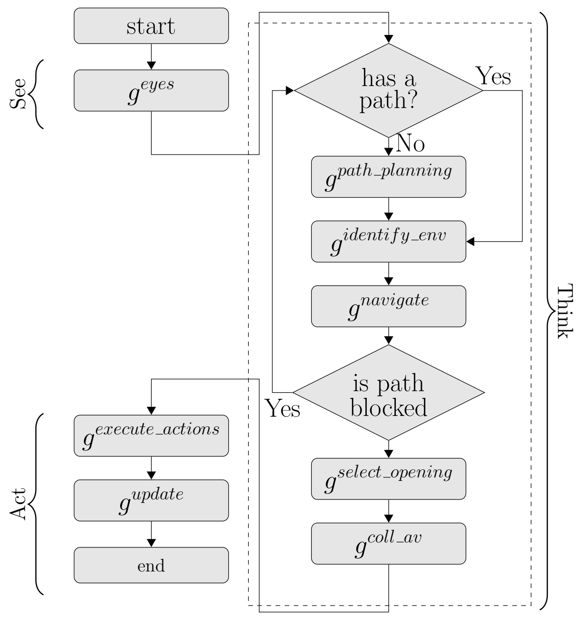


Figure 2.1: A local system update function, f_i, and a behavioral specialization example

importance for s^{ext} states as it keeps coherence between every agent's time perspective and current time-step. This model of execution allows the implementation of the dynamical system to comply with the theoretic requirements, of being parallel (from the dynamical system stand point), irrespective of serial or parallel execution (from the implementation stand point).

Chapter 3

Enhancements

3.1 Overview

The purpose of enhancing the large urban area evacuation simulation software is to enable the detailed modeling and evaluation of mixed mode evacuation scenarios. In order to achieve these goals, several enhancements encompassing all the software and model constitutive parts are implemented.

The main enhancements can be categorized in:

- Agent's enhancements
  - The inclusion of collision avoidance to represent the variation of agent's speed based on the surrounding density, base for the agents' interactions, 4
  - Automated calibration of the collision avoidance algorithm according to observed data,
  - The inclusion of car agents to represent the evacuation using cars , 3.1.1.1
  - The inclusion of car-pedestrian, pedestrian-pedestrian and car-pedestrian translational interactions based on their physical restrictions and desire to avoidance collisions, 4.5
    - Validation of the interaction models by comparing simulations results with observations, 5
  - The inclusion of other agent's interactions, such as agent's following other agents in order to model tourists or kids, agents planning their route considering the presence of other agents in order to communicate with them and help them to evacuate, etc., 5
- Environment enhancements

– The extension of the environment model in order to: provide support for the agent interactions without sacrificing the scalability of the software; provide a way to condense spatially distributed statistical data for post-processing, etc.. 3.1.2

The aforementioned enhancements made it possible to perform complex analysis of tsunami triggered evacuation scenarios considering different environmental damages and behavioral models. Examples of such analysis include the evaluation of tsunami evacuation during a large festival under full power failure considering night time conditions and the effect of implementing of emergency lighting as mitigation measures; the effect of volunteers and authorities to help and guide during the evacuation process; the evaluation of evacuations under the collapse of critical infrastructure, such as important bridges, and implementation of mitigation measures such as vertical shelters near these critical structures; the evaluation of evacuations considering different degrees of city damages as well as inundation models and their combined effects [36].

3.1.1 Specialization of agents - a^{τ}

Currently, four types of agents with simple behavioral models have been implemented to demonstrate the versatility of the framework and the use of the supporting tools provided. Although the currently implemented agents' behavioral models are simple compared to those of their real human counterparts, they are complex in comparison to the behavioral models implemented in large area simulators. It should be noted that even highly idealized scenarios can expose flaws in evacuations plans. For example, ideal agents with perfect knowledge the environment, capable of finding the best path to destinations, without considering fatigue end up drowning in the tsunami or not reaching a shelter. Thus, highlighting the need of the inclusion of mitigation measures such as additional vertical shelters, reinforcement of bridges or roads in critical paths, etc. Results of these idealized scenarios provide an upper bound, and should be improved to bring them as close to real life as possible. The software presented in this work provides a workbench to explore different behavioral models, which would ideally be supported with observation data or experts' consensus. Three types of pedestrian agents and a brief description to each type are presented below

Resident - $a^{resident}$: Represents a resident of the analyzed area. $s^{resident}$ possess a mental map of the environment, and uses $g^{path\_planning}$ to find paths according to its desired constrains and past experiences. These agents are able to store, in $s^{resident}$, their experiences such as any blocked road found, and take the gathered data into account in the decision making process. Additionally, they know the location of possible evacuation areas.

Visitor - $a^{visitor}$: Represents visitors in the considered area. They don't possess any information of the environment aside from what they can visually perceive. Their main evacuation mechanism is to seek a visible high ground or to follow other evacuees using $g^{find\_and\_follow}$. Even though they don't have mental maps of the environment, they can easily be programmed to build their own a mental maps as they explore the area. This latter functionality is essential to imitate a human visitor in a complex urban area.

Officials - $a^{official}$: This type of agents represents figures of authority, such as law enforcement, event staff, etc. Their main task is to facilitate fast and smooth evacuation by planning independently or collectively the areas to be covered by each of them using $g^{path\_planning}$, and commanding or delivering information to other agents with $g^{seek\_not\_evacuating}$ and $g^{deliver\_message}$. $s^{officials}$ also possess a mental map of the environment. A necessary item is their ability to share the experiences and update their mental maps collectively, so as to model remote communication among police officers etc. This constituent function is not included but can be easily implemented.

3.1.1.1 Cars -a^{cars}

The use of cars, as mentioned in the introduction, is the main topic of discussion in the tsunami evacuation community. To the author's knowledge, there are no models considering the detail of the interaction of pedestrians and cars in agent based evacuation simulation. In order to introduce the effect of using cars for evacuation a simplified car agent model is introduced to the simulator. This simplified model of car provides the cars with the features considered most relevant to the tsunami evacuation. Cars are assumed to know the road network of the environment, which they use to plan their path through the environment. As cars diverge from pedestrians in the degree of freedom of their movements, restriction of their movement to the road network, usage of road lanes and behavior at intersections, specialized models of navigation and collision avoidance are provided. In this specialized models cars navigate using the road network considering the different lanes and road directions, stopping at intersections, and have the ability to identify blocked roads. Additional restrictions to the steering angle of the vehicles are included to restrict their range of motion. Special care is given to the interaction between cars and pedestrians, for which field observations and specific models of interactions are introduced, see 4.3.3. The intersection model provided for the car agents allows only a single car to use the intersection (cross it) at the time, this is a pessimistic model on the use of intersections specially as the number of intersections grow, the effect of the present model of intersection as well as the car pedestrian interaction is studied in section 6.2.

3.1.2 Hybrid environment - *Env*

A detailed model of the environment is included to facilitate sophisticated sensing and behavioral models of agents, thereby overcoming the limitations of the currently used simplified models. The developed software can accommodate environments with a physical extent of 100's of km^2, in sub-meter details. The environment is modeled as a hybrid of vector and raster data, consisting of grids and graphs (see Figure 3.1) so that advantages of each of the schemes can be exploited. Model-wise the environment can be conceptualized as how an agent perceives its visible surroundings and what he knows about it. The grid represent the current state of the dynamically changing environment, including tsunami inundation and earthquake induced damages. The grid can be visually perceived by the agents with their sophisticated sensing function g^{see}, providing a base for behaviors and full use of traversable spaces.

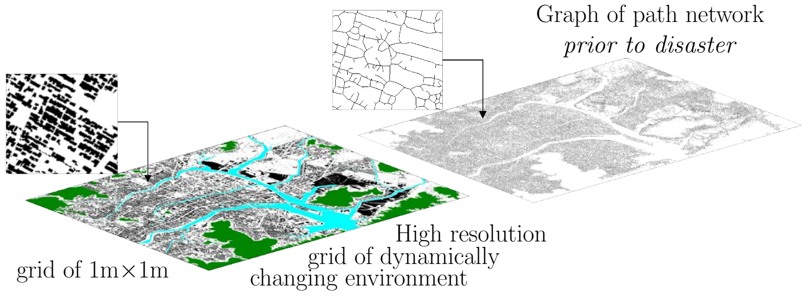


Figure 3.1: Hybrid model of grid and graph environments. Grid is dynamically updated, reflecting changes in the environment. The graph is static and represents the path network before the disaster.

In contrast, the graph represents an abstract topological model of the detailed grid environment, and included in agents' memory to provide them with the necessary geographical knowledge for their decision making. The topological graph is constructed based on the grid environment of an ordinary day. While evacuating agents experience the changes of the environment with g^{see} and discover mismatches between their knowledge, topological graph, and the current state of the environment. These mismatches and other experiences are stored with reference to this graph. These experiences include information such as if the agent received help and from whom, observed bottlenecks, etc. The graph plays a central role in agents' *think* process making it possible to implement complex decision making functions, which take advantage of their past experiences. Also, the graph significantly simplifies and reduces the computation time for process like path planning, identification of known regions, etc., which would otherwise be prohibitively computationally intensive with a grid environment. Furthermore, the graph enables compilation of statistical summaries of simulation results, see Figure 3.2. The later is especially useful for the end user of the software since it facilitates the comparison of the results of multiple simulations and

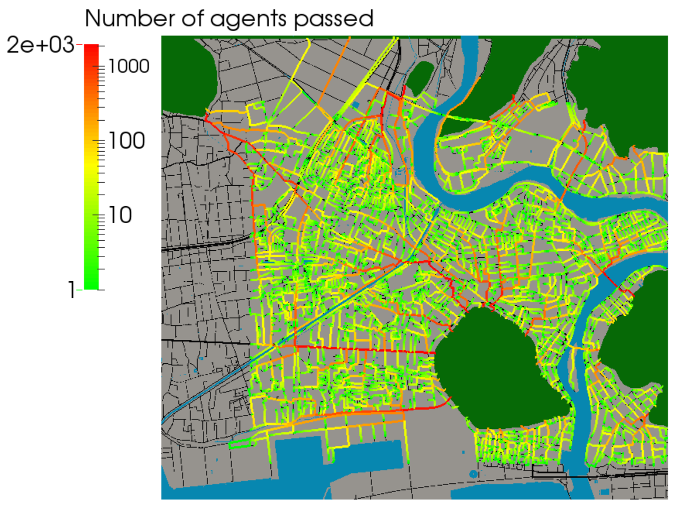


Figure 3.2: Cumulative number of agents that walked through a street segment characterized by link of the graph.

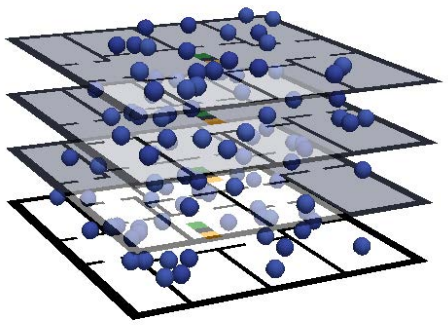


Figure 3.3: Extended 2D domain used to model vertical shelters and critical buildings

identify results such as highly used roads, bottlenecks, etc. On the other hand, comparison of the time histories of individual agents' positions in 2D grid environment is tedious and time consuming.

The use of a hybrid model of environment allows the implementation of extended 2D simulations inheriting the features of the global domain. The extended 2D domain is necessary to model vertical shelters. It requires the creation of a new data structure specifically for this purpose, building upon the hybrid environment data structure.

Additionally, distributed graph algorithms that allow to efficiently condense spatially distributed evacuation results on graphs are implemented, example in B.1. They provide a scalable way to perform path planning[4], necessary for evacuees' decision taking, e.g. find alternate routes once they identify debris blocked roads.

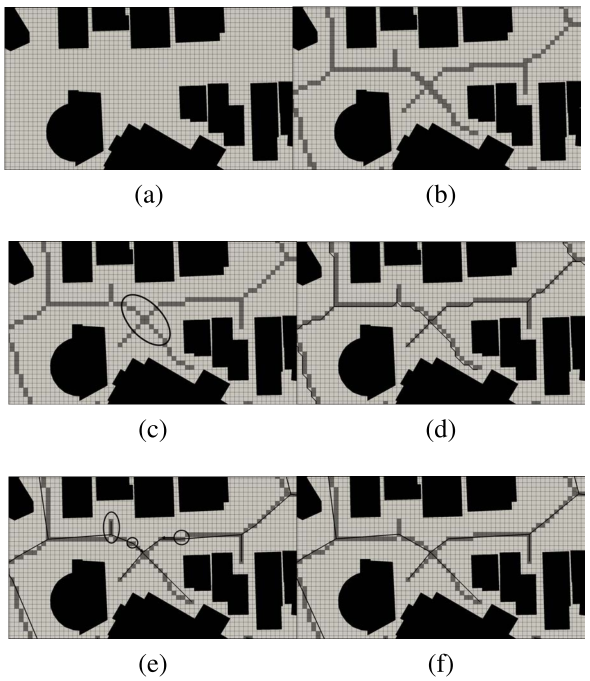


Figure 3.4: An example demonstrating the steps involved in automated generation of graph.

In order to model earthquake induced damages, $\lambda^{earthquake}$ is implemented by coupling with a physics based structural seismic response simulator called IES[5]. $\lambda^{tsunami}$ is introduced to include given tsunami inundation time history.

The manual creation of a hybrid vector/grid environment of several km in area is surely a tedious and time consuming task. To solve this problem we developed automatic model extraction tools to create the model from GIS data. The grid is constructed using standard flood fill algorithm to fill non-traversable space occupied by buildings, etc; IES has the capability to construct the grid environment either including or excluding the earthquake induced damages. Additionally, the automatic graph creation from the grid environment ensures coherence among them. The automated construction of the graphs involves the following steps:

- Thinning: Obtain a pixel skeleton connecting unoccupied spaces by pixel thinning (Fig. 3.4b); used EVG-Thin[14] by Beeson.
- Reduction of pixel skeleton: Reduce the pixel skeleton from EVG-Thin to 1 pixel width (Fig. 3.4c).

- Generation of a crude graph: Add nodes at every branching point of the pixel skeleton, at locations where the width changes rapidly, and at every 25m intervals (Fig. 3.4d).
- Prune and merge: Prune childless branches shorter than 5m, and merge the nodes within 5m neighborhood (Fig. 3.4e).
- Assign states of nodes: State of a node is set to *exit*, if there are grid cells with state of exit, within 50m visible distance.

The second step is necessary to reduce the computational cost involved in step 3. Reduction to a single pixel skeleton is automated using a set of templates for identifying and correcting parts wider than a single pixel. For the agents to navigate effectively through the damaged environment nodes at 25m interval, half of the average sight distance of an agent, are introduced.

Chapter 4

From Collision Avoidance, to interaction models

In the context of evacuation simulation quantitative analysis of the causes disrupting the evacuation need to be analyzed. Every item with considerable impacts in the evacuation time, crowd density, smoothness of flow, etc., must be adequately modeled so that field observations can be reproduced to a sufficient accuracy. The formation of crowds, the variation of speed according to crowd density, the slowing down of vehicles in the presence of pedestrians, use of all traversable space, etc., are examples of such phenomenon with high impact in evacuation time. In simplified models used by evacuation simulators, these phenomena are introduced by adjusting some factors or flow capacity. The aim of this research is to reproduce these phenomena through the emergence from agent's individual actions, and interactions, capturing their details and allowing the exploration of detailed scenarios. Experiments and field observations that capture quantitative data of these phenomena are reported in literature[32, 20, 30, 21, 31].

4.1 Collision Avoidance

A collision avoidance scheme capable of reproducing observed data is necessary. Different collision avoidance schemes have been proposed by several researchers, such as the social force model [29] and other force based models [35], quad-tree based path planning approaches [33], probabilistic maps [34], optimal reciprocal collision avoidance [17] among others. Some of these collision avoidance schemes are not created specifically to solve the collision avoidance problem but as a way to navigate avoiding collisions (as a sub-product for efficient navigation) or to specifically introduce the implicit effects among the entities avoiding collisions (social force model).Out of these schemes, the optimal reciprocal collision avoidance ORCA [17] is adopted in this project.

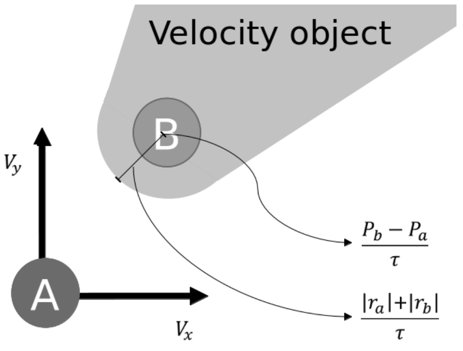
The advantages of the ORCA scheme over the other alternatives are as follows:

- the specific introduction of other moving entities and the consideration of their observed movements in the decision making process
  - in contrast to the path based models which do not explicitly consider the dynamism of the other agents require coordination or the use of the same protocol to avoid collision.
- the use of the agent's sensor data to extrapolate the movement of other agents, allowing the introduction of autonomy in the decision making on how to avoid collisions
  - in contrast to the inclusion of fictitious non measurable forces used by the force based approaches
- the consideration of all probable collision entities simultaneously (environment and other agents) to find an ideally collision free velocity
  - in contrast to simple algorithms that restrict themselves to checking the existence of a single entity in the agents path and require a conflict solver phase to chose a strategy to avoid collisions.

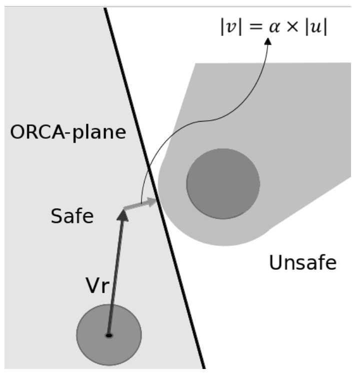
The velocities found with the ORCA algorithm are optimal only in the case of a single neighbor under the assumptions made by the algorithm. This section presents an overview of the collision avoidance scheme, implementation and exemplifies the problems of adopting the ORCA algorithm in the aforementioned multi-agent system along with the strategies to overcome them.

4.1.1 ORCA scheme

For the sake of completeness, the ORCA algorithm is briefly explained here, restrained to the two dimensional setting of interest for this project. For an in depth description refer to the original work by Jur van den Berg et. al. [17]. The physical extent of an agent is modeled as a disc with a given radius. A collision is defined as the overlap of two of such discs. In a two agents scenario, the first step of the ORCA algorithm is to identify the set of all relative velocities that bring the two agents into a collision state, which is called a velocity object (VO). Figure 4.1a shows the velocity object of B from A's perspective. Using the current relative velocity, the smallest change $\mathbf{u}$ in the relative velocity $\mathbf{v_r}$ that would bring the two agents to the critical collision state is found, see Fig. 4.1b. This critical state is reached when the distance from the outer circles of the two agents is zero, in other words both circles tangential to each other. This smallest change, $\mathbf{u}$, mark the closest tangent to the velocity object, $\mathbf{p_a} + \mathbf{v_r} + \mathbf{u} = \mathbf{V_c}$. In the original ORCA scheme a percentage of responsibility α is assumed by each pair of agents in order to avoid a collision; the



(a)



(b) ORCA plane with $\mathbf{v_b} = \mathbf{0}$

Figure 4.1: Collision avoidance overview

critical velocity change for agent A cause by agent B is considered as $\Delta\mathbf{v}_{a|b} = \alpha_a \cdot \mathbf{u}$ and the critical change for agent B caused by agent A is considered as $\Delta\mathbf{v}_{b|a} = (1 - \alpha_a) \cdot \mathbf{u}$. The line passing through that point and tangent to the velocity object is called an ORCA line. This ORCA line separates a 2D velocity domain into two half-planes. The half-plane containing only collision free velocities is called an ORCA-plane, see Fig. 4.1b. Adding the velocity change to the agent's current velocity and using the direction of the previously found ORCA-line, the ORCA-plane for the desired agent is found. Avoiding collision is then reduced to each agent choosing their velocity from the ORCA-plane.

In a multiple agent scenario, the same procedure is performed with each agent finding an ORCA-plane with respect to every neighbor. All these half planes produce a convex polygon, and the velocity is found using linear programming. The shape of the velocity objects are characterized by the geometries of the two entities avoiding collisions and the parameter τ. The quantities involved in the ORCA plane generation are: r_a is the radius of current agent; r_b is the radius of current neighbor; τ is the expected collision free travel time; and α is the amount of adjustment made by a pair of agents in a collision course as a percentage, see fig 4.1. In the 2 agent case the obtained velocity can be assured to be collision free only if the following three conditions are satisfied: 1. both agents follow the same collision avoidance scheme; 2. each agent will take the complementary responsibility percentage α of each other and the assumption of a constant relative velocity $\mathbf{v_r}$ is maintained during τ time.

As our aim is to provide autonomy to the agents, no assumption can be made over the collision avoidance scheme used by other agents or the percentage of responsibility other agents will take in order to avoid collisions. In this sense every agent takes full responsibility of its collision avoidance, i.e. $\alpha = 1$. Furthermore, in the multiple agent scenario the solutions from this scheme is not optimal and might diverge considerably from the expected human behaviors. Some of the problems of ORCA algorithm in multiple agent scenarios are described in the sections 4.1.4 and 4.1.5.

4.1.2 Implementation

Several modifications are made in order to overcome deficiencies in the ORCA scheme, to blend it with the vision based autonomous navigation scheme, and to provide more control over its functioning.

Although the authors of the ORCA algorithm propose in their paper a way to incorporate the environment in the collision avoidance scheme, it isn't suitable for the hybrid environment used in the aforementioned multi-agent system. For this purpose, a group of half-planes are created, referred int this work as navigational planes. These navigational planes are used in the same way as the ORCA planes, but with the aim to avoid collisions with the objects in the environment. The vision based autonomous navigation algorithm scans the environment in a radar like way and extracts environmental features from the information gathered. The gathered features are the openings in the visible environment and target points that provide safe references for navigation, for an in detail explanation of the process see sections 5.2.1, 5.2.2.

Two navigational planes are generated, if necessary, from the two target points from the chosen next opening and the agent's current position. In order to give the agents freedom in their mobility and to reduce the load on the linear program solver, these half- planes are moved away from the agent, using the next opening target point as pivots, ensuring that no obstacle lies inside the half-plane, and if it reaches certain threshold the half-plane is completely dropped, see B.2.

Additionally, the ORCA planes produced by the neighbors are ordered based on the distance of the neighbor. This technique allows us to relax the problem in case no feasible solution is found. To eliminate any undesired results due to the aforementioned relaxation, the navigational half-planes are given the highest priority, and the priority of half-planes of neighboring agents are set to be inversely proportional to the distance. Moreover, only the agents within a certain proximity are considered in collision avoidance, since human counterpart also does not react to all the visible neighbors. This interaction with agents in a certain proximity reduces the computational cost.

Additional parameters are introduced to fine tune the usage of the collision avoidance: a boolean *overpass* in order to introduce preferences such as if the evacuee desire to overpass ; their preferred side of over-passing, *side* (essential for cars); and restrict the minimum distances an agent prefer to maintain between other agents, ra_{pref} . Examples on the use of these parameters to implement specific interaction models are presented in 4.2. Some combinations of parameters and situations create ORCA half-planes which contradict with the agents' physical constrains (maximum acceleration, maximum speed, etc). In order to

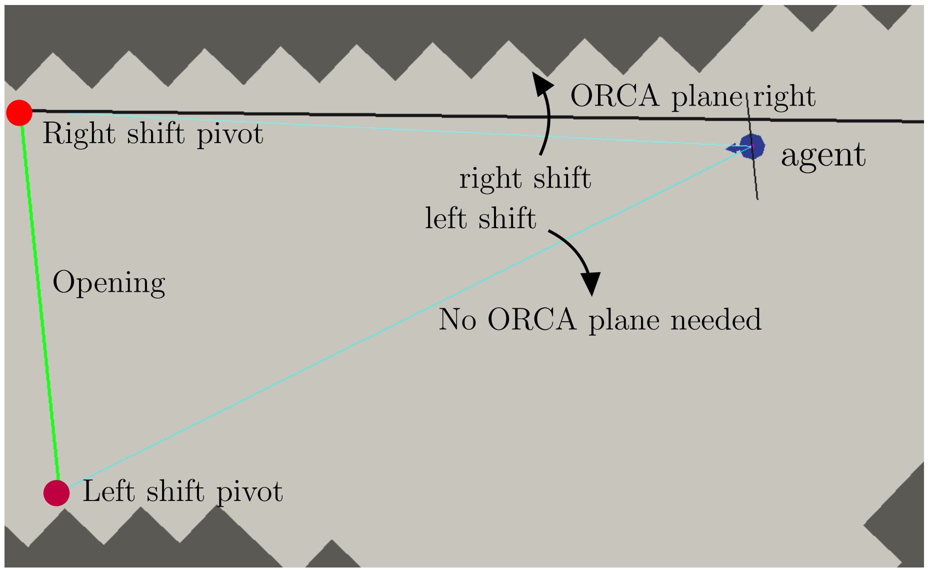


Figure 4.2: Navigational halfplanes

overcome such contradictions, every ORCA plane undergoes a simple feasibility test and the parameters are relaxed when the feasibility test is not satisfied as shown in 4.1.6.

4.1.3 Geometric development of ORCA algorithm

Although the original paper provides the general concept, geometric properties and workflow of the use of the ORCA scheme, a simple and efficient algorithm is implemented for our framework. A complete reference on the geometric properties of the VO and the ORCA itself are found in the original paper [17], in this section we will review the necessary properties needed for our implementation. The perimeter of the VO consist of 3 regions, a circular arc segment, and 2 line segments tangent to the circle describing that arc at each of the arc's ends.

Naming the agent avoiding the collision agent a and the agent to be avoided agent b and providing their respective velocities, positions and radius $\mathbf{v_a}$, $\mathbf{p_a}$, $\mathbf{r_a}$, $\mathbf{v_b}$, $\mathbf{p_b}$,$\mathbf{r_b}$ and ORCA parameters α, τ:

$$\mathbf{VOR} = \frac{\mathbf{p_b} - \mathbf{p_a}}{\tau}$$

represents the vector from the center of the a of the circular region of the VO, this vector **VOR** has the physical meaning of being the relative velocity of a needed during τ second for agent a to occupy the position of agent b. The radius of the circular region of the VO is given by $r_{vo} = \frac{r_a + r_b}{\tau}$, this having the physical meaning of being the relative speed required for any agent with radius a, tangent to an agent with radius b, moving towards the

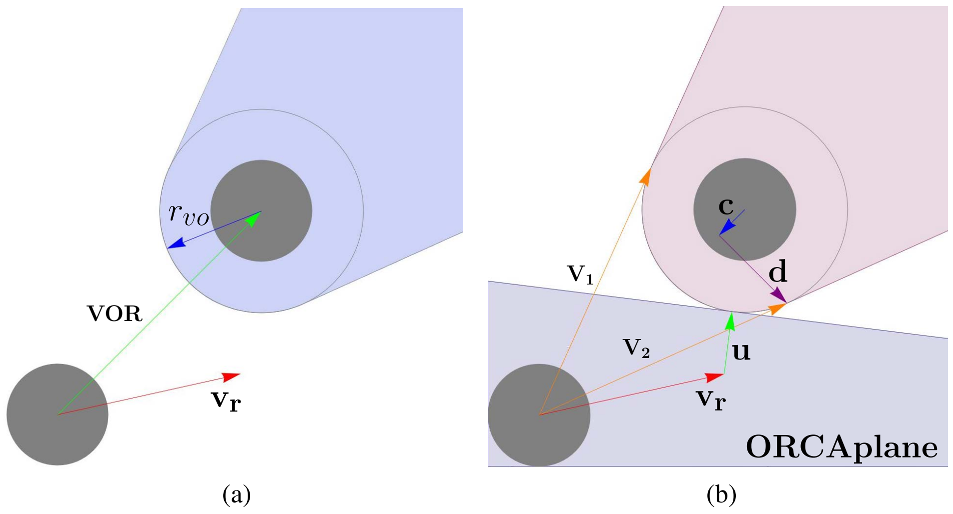


Figure 4.3: ORCA half-planes creation. Note: ORCA plane (b) with $\mathbf{v_b} = \mathbf{0}$

center of agent b during τ seconds, to occupy the position of agent b. These two parameters $(\mathbf{VOR}, \mathbf{r_{vo}})$ characterize completely the circular region of the VO, but to identify end and beginning of the arc the tangential points uniting the different segments of the perimeter of the VO have to be identified. With basic trigonometry we have that $\frac{|\mathbf{c}|}{r_{vo}} = \frac{r_{vo}}{|VOR|}$, $|c| = \frac{r_v^2}{|\mathbf{VOR}|}$ and $|\mathbf{d}| = \sqrt{r_{vo}^2 - |\mathbf{c}|^2}$. The subscripts u and n, such as $\mathbf{VOR_u}$ and $\mathbf{VOR_n}$, will be used to refer to the unitary vector and the unitary normal vector to the left,

$$\mathbf{V_{1,2}} = \mathbf{VOR_u} \cdot (|\mathbf{VOR}| - \frac{r_v^2}{|\mathbf{VOR}|}) \pm \mathbf{VOR_n} \cdot \sqrt{\mathbf{r_{vo}^2} - \frac{\mathbf{r_v^4}}{|\mathbf{VOR}|^2}}$$

These two vectors $\mathbf{V_1}$and $\mathbf{V_2}$ mark the transition between the arc region and the linear region, these two vectors have the physical meaning of being the relative velocities of a during τ seconds an agent a would require to be tangential to the left or to the right of agent b.

To identify the smallest change $\mathbf{u}$ in the a's relative velocity that would bring them to the critical collision state (the state where the two disks are tangential to each-other) it has to be identified the closest point on the VO to the relative velocity velocity of a in respect to b, $\mathbf{v_r} = \mathbf{v_a} - \mathbf{v_b}$, and the smallest change $\mathbf{u}$is the vector from $\mathbf{v_r}$ to that point.

Now with all the geometry of the VO characterized the procedure to find the critical relative velocity (closest point to the VO) $\mathbf{V_c}$and $\mathbf{V_t}$to characterize the ORCA plane geometry is shown in Algorithm 4.1. Additional parameters are introduced in this step to have better control on the creation of the ORCA half-planes; the parameter *overpass* specifies if an

Algorithm 4.1 Closest point to the VO and ORCA line direction, $\mathbf{V_c}$,$\mathbf{V_t}$

```
if ($overpass$)
    $\mathbf{v_a} = \mathbf{v_a} + (\mathbf{v_a})_\mathbf{n} \cdot side \cdot \gamma$
$\mathbf{v_r} = \mathbf{v_a} - \mathbf{v_b}$
$\mathbf{ProyV_1} = (\mathbf{V_{1u}} \cdot \mathbf{v_r}) \cdot \mathbf{V_{1u}}$
$\mathbf{ProyV_2} = (\mathbf{V_{2u}} \cdot \mathbf{v_r}) \cdot \mathbf{V_{2u}}$
if $|\mathbf{ProyV_1}|<|\mathbf{V_1}|$ and $|\mathbf{ProyV_2}|<|\mathbf{V_2}|$ then
    On arc segment
    $\mathbf{g} = \mathbf{v_r} - \mathbf{VOR}$
    $\mathbf{V_c} = \mathbf{VOR} + \mathbf{g_u} \cdot \mathbf{r_{vo}}$
    $\mathbf{V_t} = -\mathbf{g_n}$
else if $|\mathbf{ProyV_1}|>|\mathbf{ProyV_2}|$ then
    On line segment 1
    $\mathbf{V_c} = \mathbf{ProyV_1}$
    $\mathbf{V_t} = \mathbf{ProyV_1}$
else
    On line segment 2
    $\mathbf{V_c} = \mathbf{ProyV_2}$
    $\mathbf{V_t} = -\mathbf{ProyV_2}$
```

additional perturbation to v_a is to be applied, *side* chooses the direction in which this perturbation is to be applied and γ determines the magnitude of the perturbation. The values of these parameters are determined according to the interaction model to be reproduced, see section 4.2.

The smallest change in the relative velocity is then given by:

$$\mathbf{u} = \mathbf{V_c} - \mathbf{v_r}$$

This smallest change denotes the smallest change of the relative velocity that would bring both agents in a critical position (tangent to each other), as this smallest change is based on their relative velocity the factor α was introduced by the original authors to denote the percentage of this change that each of their robots were going to take responsibility for, in our case we set the value of $\alpha = 1$ as our agents take full responsibility of their movements to avoid collisions, as mentioned in the previous section.

The origin point of the ORCA-line is then given by

$$\mathbf{ORCA_{Point}} = \mathbf{v_a} + \alpha \cdot \mathbf{u}$$

$$\mathbf{ORCA_{Direction}} = \mathbf{V_t}$$

NOTE: This geometric development is restricted to the 2D case, the figures are created using $\mathbf{v_b} = \mathbf{0}$and $\tau = 1$ to ease the visualization thus making the ORCA plane tangent to the velocity object and the center of the circular region of the velocity object match the position of *b*, in general ORCA lines are parallel to a tangent of the velocity object. Conditions producing distorted geometries (such as agents in a collision state or having 0 norm vectors) require special treatment, such as aborting the creation of ORCA planes and the linear program solving, and outputting a speed of 0 and keeping the previous direction or creating half-planes with the purpose of bringing the agents back to coherent geometries.

4.1.4 Group collision avoidance

For avoiding collision with multiple agents, the original ORCA algorithm creates one half-plane per each neighbor agent and a solution is found through Linear Programming (LP). The strength and efficiency of this scheme comes from the fact that the resulting polytope (if feasible solutions exist) is convex, due to the simplification of the relative velocity space safe regions into half-spaces (ORCA-half-planes), and the existence of efficient LP solver algorithms. But, this very simplification causes severe problems. By simplifying the safe relative velocity space to a half-plane, it is ensured that the per agent optimal is present in this region hence the “optimal” in ORCA. However, the union of local optimums does not guarantee the inclusion the global optimum collision free velocity in the resulting convex polytope. To illustrate this, consider two stationary agents, B and C, positioned as shown in Fig. 4.4, and an agent *a* wants pass them avoiding any collision. Figure 4.4, shows the resulting ORCA lines and the feasible velocity regions for agent A , choosing the optimal velocity from this region at each timestep will inevitably lead it to get trapped behind agent B and C as there is no space for A to go in between them.

The best solution to this problem is to solve the non-convex optimization problem arising from considering all collision free velocities without the simplification introduced by using the half-planes; the collision free velocity sub-domain obtained by considering the union of velocity objects from each neighbor agent is convex. Another possibility is to identify this kind of problems during the building of the ORCA planes itself. Although the most desired is the first approach, solving the resulting concave problem is complex and probably time consuming.

In order to solve this problem efficiently, group collision avoidance is introduced; groups of agents among which there is no space to move through are identified and an ORCA plane is created considering this group as a single entity, see Fig 4.5d. As a simplification, 3 linear constrains form the group velocity object by identifying the farthest right, farthest left and closest agent, as seen in figure 4.5e. This although not optimal allows the exclusion of undesired solutions leading evacuees to get trapped behind other evacuees.

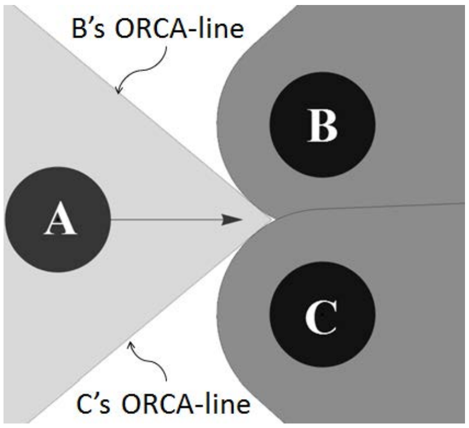


Figure 4.4: The agent chooses velocities in the light gray safe region, not being able to pass stationary agents. $\tau = 1s$, $\alpha = 1$, $p_a = \{0m, 0m\}$, $p_b = \{5m, 1.8m\}$, $p_c = \{5m, -1.8m\}$, $v_a = \{3.5m/s, 0m/s\}$, $v_b = v_c = \{0m/s, 0m/s\}$

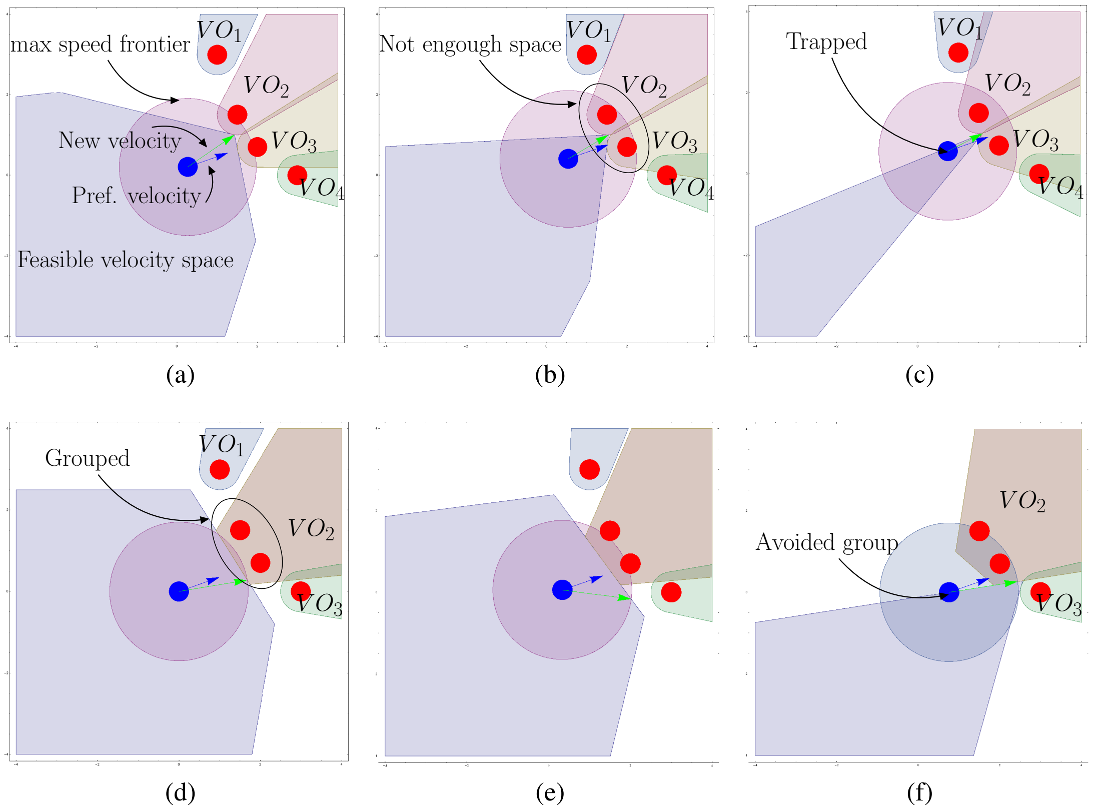


Figure 4.5: Avoiding collisions as a group

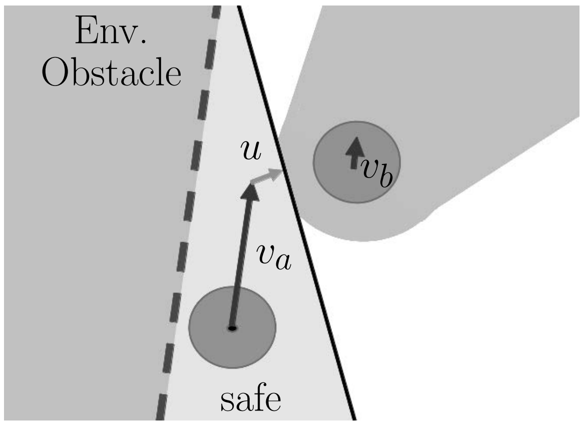


Figure 4.6: Side selection

Algorithm 4.2 Vector from current agent's position (assumed (0,0) in its local coordinate system) to the closest point on an ORCA line

```
Input: target_line (an ORCA line with an origin point and direction)
Outputs: A vector from the local coordinate system origin to the closest point on the ORCA line
return $target\_line.point - (target\_line.point \cdot target\_line.direction) \cdot target\_line.direction$;
```

4.1.5 Side selection for over-passing perturbation

In order to provide additional control over the creation of the ORCA planes, a mechanism to give priority to a certain side by adding a perturbation is introduced, as mentioned in section 4.1.3. This finer control is especially useful in order to avoid situation such as getting trapped with the environment, see Fig. 4.6, similarly to the problem described in section 4.1.4. In other to avoid this problem, it is checked whether there is a sufficient space between the agent to be overtaken and the obstacle. If there is no sufficient space in between, a small perturbation is added to the relative velocity in order to create the half-plane on the other side, if there is sufficient space on that side to overtake.

4.1.6 Feasibility check

At the time of creating the ORCA lines it is tested if this half-planes provide any feasible region, if they don't the value of τ is reduced as well as the desired velocity. Algorithm 4.4 provides means to identify if a half-plane contains feasible velocities, algorithms 4.2 and 4.3 serve as support for this feasibility check.

Algorithm 4.3 Boolean function identifying if an ORCA line contains the origin in its feasible region

```
Input: The ORCA line direction and a vector from the ORIGIN to the closest point on the ORCA line
Outputs: True if the origin is inside the feasible region of this half-plane, false otherwise
return $direction.x \cdot vectorToLine.y - direction.y \cdot vectorToLine.x < 0.$;
```

Algorithm 4.4 Function to identify if a given Halplane contains any feasible region

```
Input: The ORCA line direction and a vector from the ORIGIN to the closest point on the ORCA line
Outputs: True if the origin is inside the feasible region of this half-plane, false otherwise
Vector2D v_to_line(ORCALineToVector(target_line));
if ((!ContainsOrigin(target_line.direction,v_to_line))&&(v_to_line.R()>max_speed))
    return false;
else
    return true;
```

4.2 From collision avoidance to interaction models

The inclusion of collision avoidance alone is not enough to model the complexity of human behavior in the simulator, even more if we consider further deviations necessary to model vehicles. In order to have finer control over the way agents take decisions regarding how to avoid collisions, further parameters are introduced. This section introduces the way these parameters are chosen or manipulated in order to represent specific behaviors. Although the rules and decision models (how the agent chooses its parameters) presented in this section are simple, it will be shown that they enable our simulation result to be in good agreement with the observed behaviors.

4.2.1 Pedestrians

Pedestrians are modeled to consider asymmetrically their interactions with other agent on their back and agents on their front, as it is assumed that their awareness of their need to actively avoid collisions with other agents is defined with what they can see and consider close, while their awareness of other agents on their back is based on their sensing of the other agents through sound or other features they can sense. Sound and other senses are not especifically included but modeled using this asymetry. In order to model pedestrians preferences in different density situations, pedestrians in tight crowds will consider different collision avoidance parameters than those in sparse situations. Parameters such as the distance to be kept to other pedestrians ra_{pref}, the time they expect other pedestrians to keep the same velocity τ, etc. Special care was given to the interactions between cars and pedestrians, as pedestrians actively try to get out of the cars trajectory if felt in danger. The specialized algorithm to include pedestrians‘ preferences in the way they avoid collisions is presented in algorithm 4.5

4.2.2 Cars

Just as the pedestrians, cars are modeled to avoid collision with whatever lies in a certain visible proximity. Default values and related logic for setting the parameters, which influence the movement of cars, are shown in the algorithm 4.6. Car agents could be further improved by setting τ according to the surrounding pedestrian density and speed, instead of the constant value of τ set at the instantiation.

In order to constrain a vehicle's motions to its current lane, half-planes are created to demarcate the traversable region of the lane. An additional set of half planes restrict the maximum steering angle of vehicles. Cars wait for their turn to continue their path at intersections and one car at a time is allowed to move through the junction. Though this rule can model junctions of one lane roads, it lowers the flow in other roads. Especially, in multi lane roads, this simple treatment lowers the flow significantly. While waiting for their turn, cars are not included in the neighbor list of the neighboring pedestrians, so that they can pass through the cars. This models pedestrians crossing the road in between the spaces of stopped vehicles or along pedestrian crossings.

4.3 Specific interactions models $g_{c-p}^{coll\_av}$,$g_{p-p}^{coll\_av}$,$g_{c-c}^{coll\_av}$

The previous two sections provide general models to introduce pedestrian and cars collision avoidance preferences to the model, they require an initial value for the collision avoidance parameters as well as values to instantiate the constants used in the algorithms. These preference parameters and constants are then dependent of the type of interaction being modeled. A specific instance of the preference parameters and constants is required to match an observed behavior. A specific set of parameters and constants to reproduce a given behavior and their respective algorithms, 4.5,4.6, is what represents a specific interaction model. The process of finding the specific parameters is what will be called tuning or calibration. The result of this interaction models is the emergence of relationships between the amount of evacuees interacting within certain visible proximity and the evacuees speed. Following sections present the different specific interaction models and the emergent relationships arising from them contrasted with observed data. It is important to note that the interaction models presented in this section are calibrated with observations not taken during evacuation situations. The aim is not to provide a specific instance of the parameters but to show that given observation data different interaction models can be created through the right choice of parameters. Appendix C shows the set of parameters producing

Algorithm 4.5 Default values and rules for setting parameters which influence pedestrian movements.

Input: $real\_ra$, real radius representing an agent occupancy, $comfort\_ra$, radius representing the comfortable space of the agent, va, desired velocity, τ parameter of the collision avoidance scheme, overpass, side parameters defining the over-passing maneuver, $neighbor\_list$, list of evacuees pedestrians and cars visible to the agent.

Output: ra,va,τ,$overpass$,$side$

```
//
$closest\_agent = FindClosestAgent(neighbor\_list);$
$closest\_car = FindClosestCar(neighbor\_list);$
$close\_agent\_min\_distance(12);$
$close\_car\_min\_distance(5);$
$crowd\_high(12);$
$crowd\_mid(7);$
$tau\_factor(2);$
//Aid the overpass slow close agents going in the same direction
if(($RelativeDistance(closest\_agent) < close\_agent\_min\_distance$) and ($closest\_agent.speed < my.speed$) and($SameDirectionOrStop(closest\_agent)$))$\{$
    $va = 10 \cdot va$; //exaggerate preferred velocity
    $overpass = true;$
    $ChooseSide(side, closest\_agent);$
$\}$else if($RelativeDistance(closest\_car) < close\_car\_min\_distance$)$\{$ //Get out of the way of cars
    $va = 10 \cdot va$; //exaggerate preferred velocity
    $overpass = true;$
    $ChooseSide(side, closest\_agent);$
$\}$
//Choose parameters based on crowd density
if($neighbor\_list.size() > crowd\_high$)$\{$
    $ra = real\_ra;$
$\}$else if($neighbor\_list.size() > crowd\_mid$)$\{$
    $ra = real\_ra * 1.1;$
$\}$else$\{$
    $\tau = tau\_factor * \tau;$
    $ra = comfort\_ra;$
$\}$
```

Algorithm 4.6 Default values and rules for setting parameters which influence car movements.

```
Input: $va$, desired velocity, $\tau$ parameter of the collision avoidance scheme (currently not used), overpass, side parameters defining the over-passing maneuver, $neighbor\_list$, list of evacuees pedestrians and cars visible to the agent.
Output: $va$,$overpass$,$side$
//
$closest\_agent = FindClosestAgent(neighbor\_list)$;
$closest\_pedestrian = FindClosestPedestrian(neighbor\_list)$;
$close\_agent\_min\_distance(12)$;
$far\_distance(10)$;
$close\_distance(5)$;
$speed\_factor\_far(0.1)$;
$speed\_factor\_close(0.5)$;
//Overpass slow close agents going in the same direction
if(($RelativeDistance(closest\_agent) < close\_agent\_min\_distance$) and ($closest\_agent.speed < my.speed$) and ($SameDirectionOrStop(closest\_agent)$)){
        $va = 10 \cdot va$; //exaggerate preferred velocity
        $overpass = true$;
        $ChooseSide(side, closest\_agent)$;
}
//Disable overpass if
if(($RelativeDistance(closest\_agent) < close\_distance$) and ($my.speed < my.max\_speed \cdot speed\_factor\_close$)){
        $overpass = false$;
}else if(($RelativeDistance(closest\_agent) > far\_distance$) and ($my.speed < my.max\_speed \cdot speed\_factor\_far$)){
        $overpass = false$;
}
//If pedestrian closer than 0.5m cars reduce their velocity or stop
if(($DistanceToColision(closest\_pedestrian) < 0.5$)){
        $va = 0.1 \cdot va$;
        $overpass = false$;
}
```

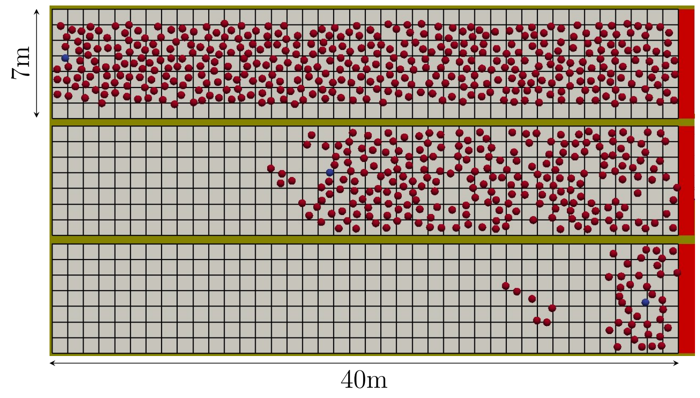


Figure 4.7: Three snapshots of a simulation conducted for validating pedestrain-pedestrain collision avoidance. Pedestrian-pedestrian validation setting, target agent in blue.

the interactions presented in this section, along with the limits used for the tuning described in section 4.5.

4.3.1 Pedestrian-pedestrian interaction - $g_{p-p}^{coll\_av}$

As most human behavioral processes, observed crowd movements present data with high variability, as human behavior varies from every individual to another, and even by considering the mean behaviors these vary with age groups, culture and even time of the day. Common observation data are presented as relationships between a representative individual of the crowd, or the crowd itself, property and a parameter representing the configuration of this crowd, usually crowd density. In this section the pedestrian-pedestrian interaction model is validated by comparing the results of the simulations with field observations reported by Mori et al. [20], and with a regression over various observations compiled by Weidmann [21]. The observation data from Mori et al. presents observations of unidirectional flows observed through bird eye camera in Osaka in walkways ranging from 2.2 m to 4.5 m in width and 20 m in length and an open boundary condition, while Weidmann data as a major compilation of pedestrian research aggregates data from 25 research publications with different boundary conditions, different flow directions and varying widths.

The validation scenario consists of a 5m road plus 1m on each side to simulate the effect of the open boundaries and 40m in length. A target agent is positioned among a crowd, with a given maximum speed. Then this agent and its surrounding crowd are set to evacuate in an unidirectional flow, and its average speed is recorded, see Fig. 4.7).

Using this setting, 25 Monte Carlo Simulations (MCS) are performed; one for each starting crowd density. Each MCS consisted in 100 simulations each. The average speed of the target agent is recorded for each simulation and its distribution is presented as a whisker-box plot per MCS along with a polynomial regression performed to data by Mori

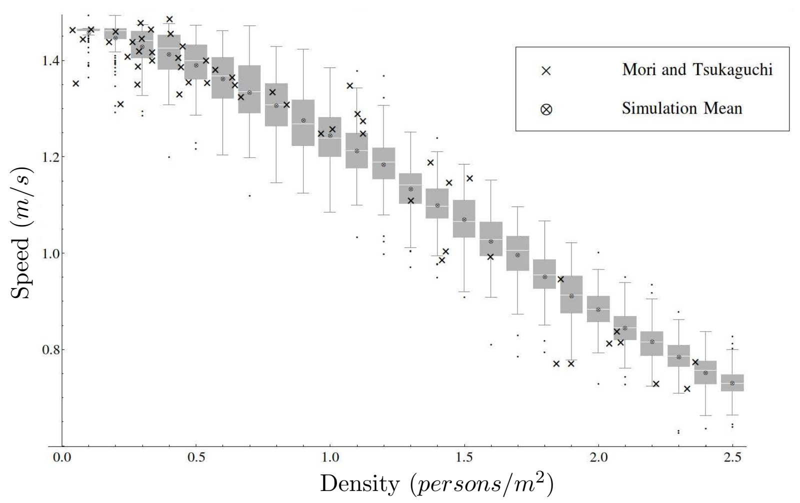
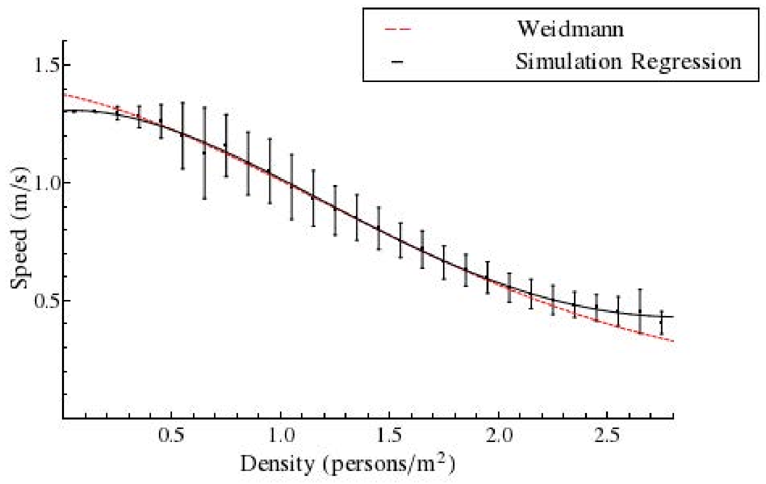


Figure 4.8: Whisker plot of the validation results against Mori and Tsukaguchi observations. Outliers considered as further than 1.5 inter quartile range are plotted as black dots

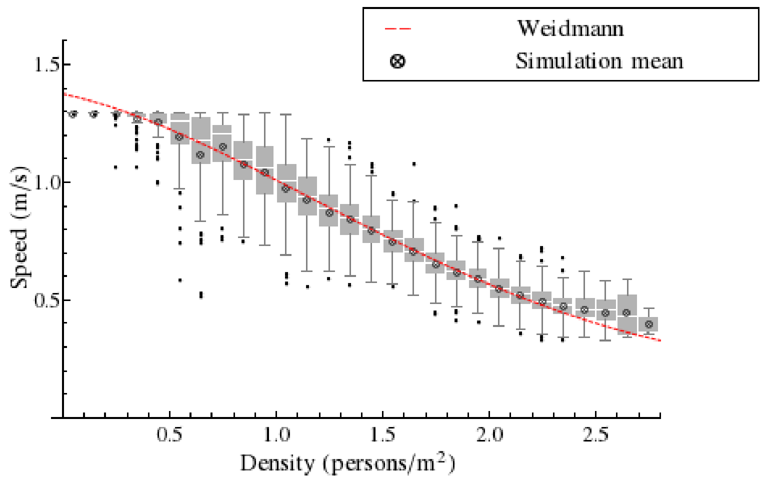
et al., see Fig. 4.8. As is seen, the simulation results are in good agreement with the observations, especially up to the density 1.0.

Using the same validation scenario and re-tuning the parameters, the simulations results are compared with the observations reported by Weidmann [21]. Figure 4.9 compares the numerical results with the reported observations. In this case is impossible to compare the dispersion of the data with the dispersion of the data produces by the simulations as only the regression over the data points are provided. It can be observed that the average behavior produces a good match with the regression reported by Weidmann. Additional fitting could be performed by introducing finer tuning capability in the parameter choosing phase, by making further subdivisions in what the agent considers different crowd sizes in algorithm 4.5, or by introducing a sense of available space in relationship by the number of surrounding agents instead as the plain number of surrounding agents. For the purpose of the simulation these results are considered to provide sufficient agreement with the observation data. For more specific pedestrian-pedestrian interactions to simulate scenarios more sensitive to this interaction, different flow directions and boundary conditions should be tunned independently for the purpose or/and “match all” calibration parameters should be found.

The objective of this study is to identify effective strategies to smoothen the evacuation process. For this reason, the validity of the implementation of the collision avoidance algorithm for densities higher than 2.5 $people/m^2$ isn't checked; any evacuation strategy producing high density crowds is out of the scope as any evacuation with such densities is already dangerous.



(a)



(b)

Figure 4.9: Validation with Weidmann regression. (a) Regression over the results and standard deviation as error bars, (b) Detail of the simulations presented as error bars and outliers (>1.5 times IQR)

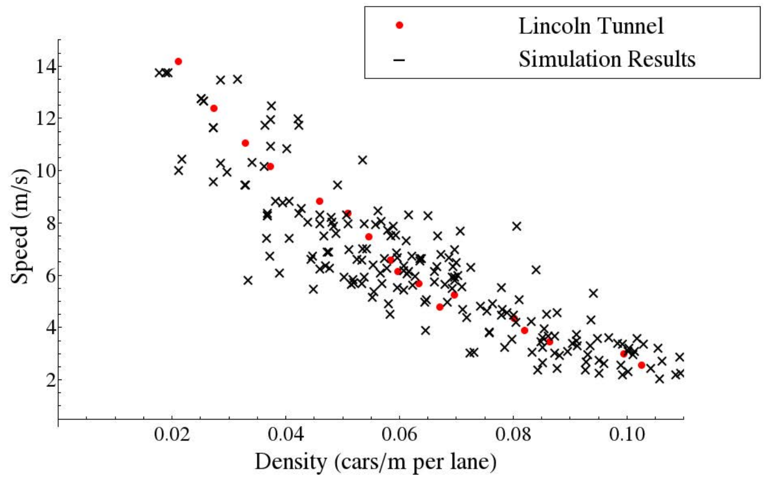


Figure 4.10: Car-Car interaction validation results

4.3.2 Car-car interaction - $g_{cc}^{coll\_av}$

The validation of car-car collision avoidance is performed by comparing the simulation results with Lincoln tunnel observations [37]. Movement of car agents along a single lane of 1km length is considered as the problem settings. 100 simulations with different initial densities are performed. Samples of matching lane densities and current speed are obtained every 0.2s and the average over 20s is reported as a data point. As seen in Fig. 4.10, the simulation results are distributed around the observation data. While most points lay within 0.5m/s distance from the observation data, some of the simulation results have 2m/s disagreement with the observations. These outlying points can be reduced by further tuning. In the future, validation should be conducted against observations with a large number of records.

4.3.3 Car-pedestrian interaction on narrow roads- $g_{c-p}^{coll\_av}$

4.3.3.1 Field observations

In contrast to the validation of the above presented interactions there is no available observation data that could be used for the validation of the interaction among cars and pedestrians. Because of this field observations are performed from 2 locations near Nezu shrine, Tokyo, Japan during golden week. Videos of 72 cars passing through the surrounding crowds are recorded with a normal video camera (Sony Handycam HDR-CX170, 1080i). The aims of these observations are:

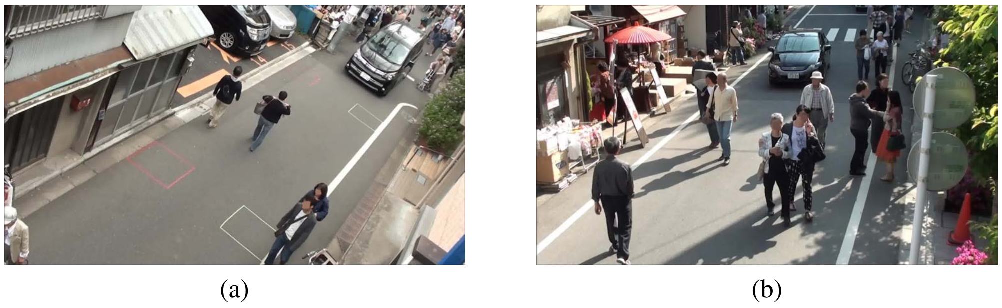
- Observe common behaviors of pedestrians and cars when interacting in a close proximity, in order to decide rules for the car-pedestrian interaction model

- Find a quantitative relationship between pedestrians and cars for the validation of car-pedestrian interactions

The videos are recorded using a 3m tall pole, at 2 locations. Fig. 4.11a shows a frame of the videos taken from the second floor of a private house next to the street, from now on referred as area 1. The number of observation from this position is limited. The second batch of measurements are taken by the edge of the road, holding the pole on the pavement. The observed area is measured, and marks are made on the asphalt in order to have reference measure points to quantify the movements and interactions (the squares, 1m$\times$1m, in Fig. 4.11a. Two reference points marked the start and the end of analysis area. The distance between these reference points is 12.25m for area 1 and 19.7 or 18.22 (depending on the visibility, as these marks were sometimes occluded from the street pole view). These practical restrictions constrained the quality of the videos taken, and the possibility to use commercial software for the automatic trajectory extraction. The results in this section do not provide a valid model for interactions during a tsunami evacuation as the measures are performed during a festive day with a majority of tourist under no stress condition. Given a similar video recording of people evacuating from a tsunami, etc., same procedure given below can be used to quantify and tune the model to imitate their behaviors. Surely the quality of the videos collected are not the best to attain the aforementioned two objectives. However, the main points of this section are to 1: show the possibility to quantify the interactions between pedestrians and cars; and demonstrate how this data could be incorporated to the model and reproduced through the tuning of the relevant parameters.

The first step in analyzing the video recordings was to trim down the videos and extract the moments when the cars entered the analysis area and left it, since most of the videos consisted of frames without cars. Once the videos of the individual cars were trimmed, frames at constant intervals were extracted. Although several automatic trajectory tracking techniques have been reported in literature [add references], most of them cannot handle video recordings at an oblique angle to the road surface. In order to perform a manual analysis of the data, two sets with different granularities are created. The first set is created with frames at 1s interval (1,496 frames) and the second set is created extracting frames at $0.1s$ intervals (14,757 frames). From the 0.1s frame batch, for each car the starting and the end frame are recorded. In the coarse grained frame batch for each car the number of pedestrians in the vicinity Ω_t are counted.

Based on the available data, the quantities C^v and S^v are defined to quantify the video recording observations from the vehicles' perspective. C^v characterizes the state of the "interacting" neighborhood of a vehicle (consisting of other evacuees and the environment) during its movement through the observation area. While S^v characterizes the "behavior" of the vehicle while moving through the observation area. As the physical environment reminded constant throughout the observations it doesn't appear in the construction of C^v, definition of these quantities are given as follows.



(a) (b)

Figure 4.11: Sample frames extracted from the observations examples

4.3.3.2 S^v

Using the first frame a vehicle entered the observation area and noting the last frame when the vehicle left the “observation time” the time a car took to cross the observation area was determined, and dividing it by the distance between the two marks defining the starting and ending region, observation length, the parameter S^v is obtained. This parameter is close to the average speed of the car in the observation area as the restriction of car being on the lane constrained the directions changes that would cause major divergences from average speed.

$$S^v = \frac{\text{observation length}}{\text{observation time}} \approx \text{average speed}$$

4.3.3.3 C^v

As the observations are discretized in T number of time-slices (different for every observation as by extracting frames every Δt seconds, as noted before and by defining the region where the presence of pedestrians would influence the behavior of the vehicles as Ω_t, from now on called "interaction" areas, and counting the number of pedestrians appearing in that region.

$$f_t(p) = \begin{cases} 1 & \text{if } p \in \Omega_t \\ 0 & \text{if } p \notin \Omega_t \end{cases}$$

where p represents the position of a pedestrian in the surrounding of the car. The difficulty comes in the definition of Ω_t as no quantitative parameter is available to perform the definition of the shape and extent of this region. Through the observation of the video recordings it was estimated that cars started slowing down when pedestrians were located at 3m or less in the road in front of them, it was also identified that the cars never left the road are delimited by the white lines, using these observations as estimations the interaction region is defined as a rectangle with a width equal to the lane size and a length equal to $3m$

Cumulative number of pedestrians second in the interaction area is given by.

$$C^v = \sum_{t=1}^{T} \sum_{i=1}^{n} f_t(p_t^i) \cdot \Delta t$$

The counting was performed manually, so the total count per frame was saved in a vector.

$$F_t = \sum_{i=1}^{n} f_t(p_t^i)$$

the expression is then reduced to

$$C^v = \sum_{t=1}^{T} F_t \cdot \Delta t$$

Data obtained for each car is aggregated into a graph without visible differences between the points coming from the different observation regions, which means that the differences in the two regions were not significant enough as to be distinguishable with the amount and quality of the data collected (see Fig. 4.13).

Given that f_t only takes the values 0 and 1 it follows that $0 \leq F_t \leq n$, then $\lim_{\Delta t \to 0} C^v = \int_1^T F_t dt$ converges. This highlights the robustness of the measure imposed on the state of the neighborhood, C^v. It is important to note that there is a lack in robustness regarding Ω_t . The rectangular shape of Ω_t is chosen based on the ability to visually estimate the shape based on the road geometry, environment features such as doors and known distances. The ability to perform an easy visual estimation is a requirement for the manual extraction of the data. Various dimensions of Ω_t are evaluated and the one presented in this section is the one producing the best results. Better defined shapes, with unique dimensions, would improve the robustness of the estimator but would also require the use of computer vision techniques to evaluate and extract the data.

4.3.3.4 Validation of car-pedestrian interaction

Cars and pedestrians parameters are tuned, and the same measuring technique described in sections 4.3.3.2, 4.3.3.3 is used. Figure 4.12 provides a schematic view of this process. 100 simulations with different initial densities are performed, $C^v$is constructed with a $\Delta t = 0.2s$ over the time, T, required for the vehicle to cross the interest are with its matching S^v is reported as a data point for each simulation, see figure 4.10

Figure 4.13 shows that both the simulation results and the observed data follow a similar trend and have a similar dispersion, indicating that those are in good agreement. With high quality data, from which more details can be extracted, better indices to quantify the car-pedestrian interaction in narrow roads can be defined. Such high quality data would make

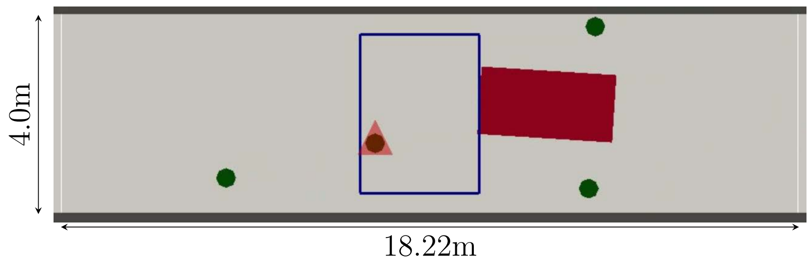


Figure 4.12: Car counting

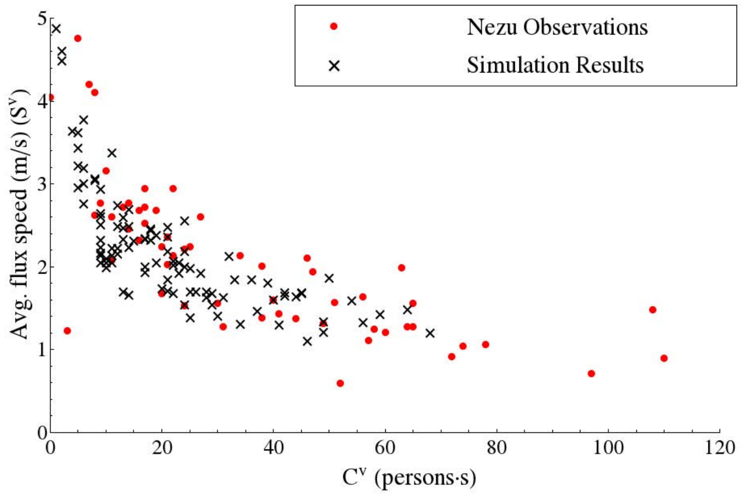


Figure 4.13: Car-pedestrian interaction validation results

it possible to fine tune the model. Though this tuned model is not suitable for mimicking tsunami evacuation scenarios, this model is used in the chapter 6 due to the lack of data.

4.4 On the validation

Although these interactions have been validated individually, there are conflicting parameters between them which brings out the question as to which set to parameters use for the complete scenarios. To solve this problem several approaches can be taken; decide which interaction requires the most accuracy and prioritize their parameters; as the parameters producing good agreement with the observations are not unique and are bound to the threshold determined to decide if the agreement is good, several iterations of tuning could be performed as to find a set of parameters that satisfy best all the interactions simultaneously. Furthermore the major roadblock encountered during these validation were the vague definitions of density used, as densities vary in space, a proper definition of this locality of the measure is needed, work on the definition of more robust density definitions use voronoi tessellations to increase the robustness of these measures, [31].

4.5 An automated parallel calibration/optimization tool - g^{coll_av}calibration

Given that evacuations during tsunamis involve a large number of lives, it is of primary concern to improve the reliability of the simulation results. Human behaviors change according to the situation, time of the day, area, country, etc. It is highly unlikely to have a model with a unique set of parameters reproduce all the possible situations. The best solution is to provide a versatile tool for calibrating the evacuation simulator to the target population, using field observations. For this purpose an automatic calibration tool is developed. The evacuation simulation software is then used as a mapping from the parameter space to the observation space. The search ranges in the parameter space have to be properly restricted with reasonable upper and lower bounds for each parameter, excluding unfeasible regions. These bounds are usually based on physical restrictions, e.g. speed should be between zero and running speed. Then a measure of the disagreement between the evacuation simulator results and observations can be casted to a suitable error norm, and the problem of optimum calibration can be reduced to an error minimization problem. As an example, given a regression on the observed data or a fundamental diagram, $r(x)$, and n simulation data points (y_i,x_i) in the observation space, the optimal calibration can be found by minimizing the L^1-norm defined as $\sum_{i=1}^{n} |r(x_i) - y_i|$. Another useful measure is L^∞-norm which concentrates on the reduction outliers.

The finding of an optimal calibration is reduced to minimizing the error. Other measures can be used, for example a calibration using an L^∞ norm that would focus on reducing the number of outliers.

4.5.1 Search strategies and parallelization

This search for an optimal calibration is a combinatorial optimization problem, for which a parallel meta-heuristic trajectory based approach is considered. Different approaches can be adapted to explore the search space. Currently, two approaches are implemented in the automated calibration tool. The first is a progressive search which introduces small random perturbations to the parameters at a small neighborhood of a given parent point, and move to the parameter set that has the lowest error. If no neighboring point with a lower error is found after a predefined maximum number of random perturbations, more aggressive perturbations are applied and the search is restarted at a new point. The second approach is based on simulated annealing, which inverts the former process by starting with more aggressive perturbations and reducing the level of aggressiveness as the search progresses.

A schematic view of the search process is seen in Figure 4.14.

This process is computationally intensive as for every trial parameter set a complete simulation has to be performed. Furthermore, in order to assure that the simulation is not being fitted to an outlier/special case scenario Monte Carlo simulations are performed. Monte Carlo simulation and the presence of many independent search fronts increase the parallel vocation of the problem. The automated tool utilizes the Message Passing Interface (MPI) to communicate and spawn the processes. Figure 4.15 shows the search with 2 levels of paralellization.

4.5.2 Other applications

The parallel calibration tool is designed to attach to any client application, with minimal requirements and changes for the client application. The client application, such as the presented evacuation simulator, would ideally use the Message Passing Interface (MPI) to communicate back with the calibration/optimization tool, although other models of communication are possible.

This tool is useful to accelerate the assessing of the versatility of the different agent based tools. Answering questions such as if the current model is enough to reproduce the observed behavior or if further improvements are needed in order to capture the essence of the phenomena and ultimately discovering features or lack of them in the models .

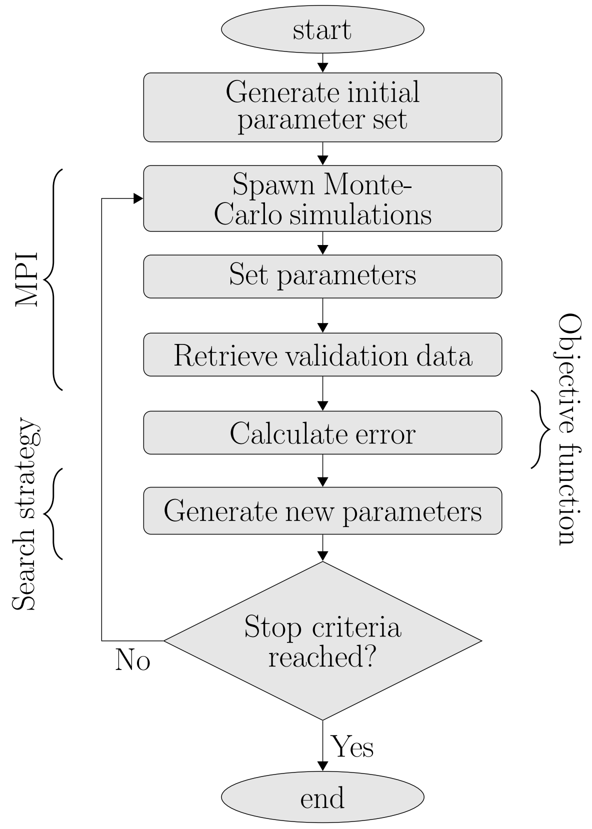


Figure 4.14: Flowchart of the automated calibration tool.

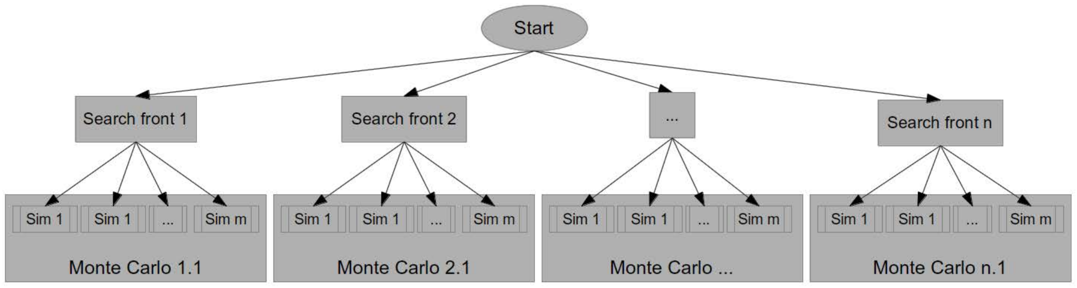


Figure 4.15: Two layers of parallelism (search and Monte Carlo parallelism)

Chapter 5

Constitutive functions of agents and environment

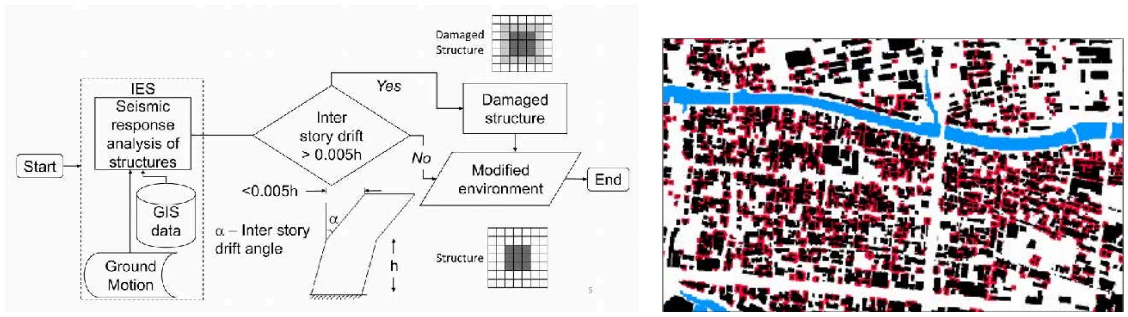
This section provides a deeper insight at the environment update functions λ and some of the agent's constitutive functions g

5.1 Environment update functions - λ

5.1.1 Modeling of tsunami inundation $\lambda^{tsunami}$ and earthquake damaged environment $\lambda^{earthquake}$

Accurate representation of earthquake induced damages and tsunami inundation is needed to take advantage of the detailed models of environment and sophisticated agents, thus overcoming the constrains of simplified agents and providing access to a larger range of scenarios to be explored with finer control over them. The Seismic Response Analysis tool (SRA) is used to generate the model of the damaged environment as seen in figure 5.1. In order to introduce tsunami inundation models, the grid environment is updated with inundation data.

The advantage of the use of a hybrid model of environment is highlighted with these dynamic updates of the environment; only the grid needs to be updated as it represents the current state, damaged or inundated, of the environment; the graph allows the agents to experience incongruences of the real state of the environment with their previous knowledge of it and store this experience in an efficient manner in respect to the graph.



(a) (b)

Figure 5.1: Damaged environment (right) generated according to physics based seismic response analysis (left); shown in black and red are space occupied by undamaged buildings and debris

5.2 Constitutive functions - g^{exp}, Vision, navigation, path finding

5.2.1 Vision-$g^{eyes} + g^{identify\_env}$

g^{eyes}extracts information from their visible environment, N^{env}, creating a visual boundary in s^{int} based on an agent's sight distance. Analyzing the information extracted from vision in s^{int} , the agents identify, $g^{identify\_env}$, features necessary for navigation, such as obstacles and available paths. The agents' vision possess functionality to scan the grid environment like a radar and identify the boundary of its visible horizon. This process is computationally expensive, because of this the radar recognition has been implemented as a templated loop-up table in order to avoid solving a large number of linear equation during run time. Additionally this template allows the definition of multiple origins in order to reduce the error introduced by using precalculated radar rays not originating exactly from the agent's position and the template with the origin closer to the agent's current position is used to look up the rays. The information of the boundary of visibility is saved in a circular container (i.e. circular linked list or a STL vector with a circular iterator), which holds what lies at the end of each ray: an obstacle, exit or open space. Furthermore, a list of visible neighbor agents is created. Analyzing this information, three main features, *exits* , *open paths* and *probable paths* are identified, see Fig. 5.2. Regions in the horizon, where the radar rays reached open spaces are marked as open paths. Probable paths are identified in regions where no open paths are found on the horizon and adjacent rays that reached obstacles largely differ in distance, here on called sudden jumps. Figure 5.3a gives an example of these identified features.

Correct identification of these features is essential for navigation since the saw toothed nature of the grid environment also produces sudden jumps similar to those produced by

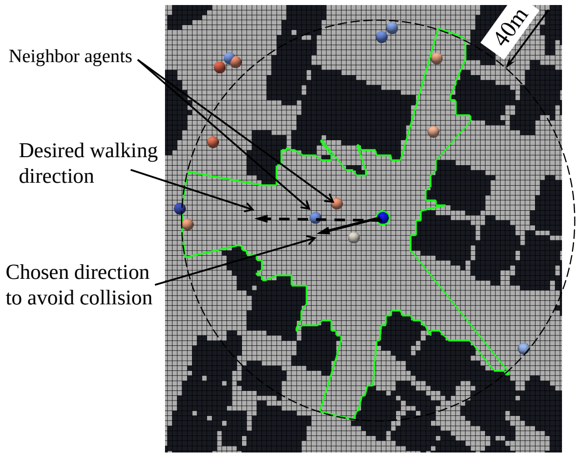

Neighbor agents
Desired walking direction
Chosen direction to avoid collision
40m

Figure 5.2: Agent horizon, and identified features

the roads at a junction. Letting the agents move too close to those false identified features make the agents' behavior unrealistic or make them get lost. Due to the radar like nature of the vision, an agent getting too close to a corner or edge obstructs its vision. Sudden jumps in its vision could be incorrectly classified as probable paths or no paths could be identified thus trapping the agent or forcing it to turn around and try to find another route. Because of this as part of the features identification, safe vision upper bound points called targets are identified 0.5m away from the closest obstacle along with the openings, see Fig. 5.3b. These bounds are not artificial; there is always at least half the shoulder width distance between an obstacle and the eye of a person.

5.2.2 Navigation -$g^{navigate}$

The agents' navigation defines the way the information collected using the vision is going to be used to dynamically find a route the the desired destination. This information is analyzed and based on the priority of each of the openings, the opening with the most likelihood to lead the agent to its desired destination is chosen, here on referred as *next opening*. The *next opening*, the vector containing the identified neighbors and the desired direction are then used as input for the collision avoidance algorithm to produce an ideally collision free velocity for every agent, see Algorithm 5.1.

5.2.2.1 Navigation for Non-Visitor agents - $g_{nv}^{navigate}$

Using their knowledge of the environment, the non-visitor agents can find a route to a desired destination. The paths are found using the Dijkstra's algorithm on the agents' mental map of the environment and the paths are defined by nodes of the graph or a reduced

Algorithm 5.1 Autonomous visual based navigation

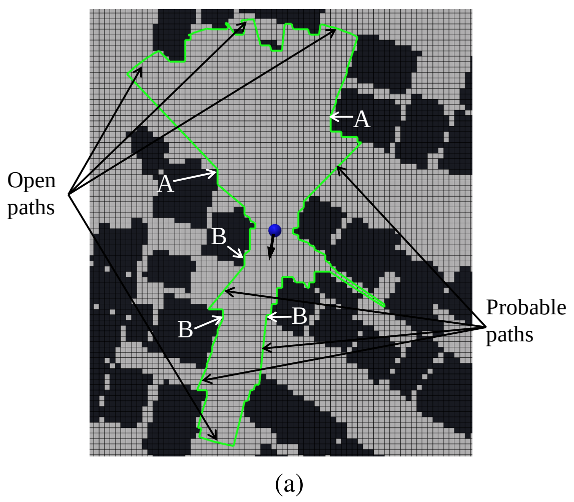
```
input  : Agents properties, desired moving direction $\mathbf{v}_d$ (far visible destination or landmark direction)
output: Next moving direction: $\mathbf{v}_n$
/* vision.IdentifyFeatures() Processes the information available in the vision object, and
outputs vectors of open/probable paths and identified exits                              */
vision.IdentifyFeatures(front/back_open_paths, front/back_probable_paths);
next_opening.clear()
while not Frozen do
    if front_exits.size() > 0 then
        next_opening ← ClosestExit(front_exits,$\mathbf{v}_d$);
        front_exits.clear()
    else if back_exits.size() > 0 then
        next_opening ← ClosestExit(back_exits,$\mathbf{v}_d$);
        back_exits.clear()
    else if BestPath(next_opening, front_open_paths, front_probable_paths,$\mathbf{v}_d$,min_jump) then
    else if BestPath(next_opening, back_open_paths, back_probable_paths,$\mathbf{v}_d$,min_jump) then
    else                                                          /* agent has no paths */
        Freeze the agent
    end
    if not next_opening.empty() then break
end
if not FindVelocity($\mathbf{v}_n$,next_opening, Agent Properties) then Freeze the agent; /* No satisfactory $\mathbf{v}_n$
found */
return $\mathbf{v}_n$
```

Function BestPath

```
input  : next_opening, open and probable paths in the font or back; open_paths, probable_paths, $\mathbf{v}_c$,
         $\mathbf{v}_d$,min_jump
output: Whether the openings have been exhausted and the next_opening if found
/* BestProbablePath() and BestOpenPath() find the probable and open path closest to the
direction of destination, avoiding narrow paths                                         */
dot_max = dot_max_p ← -1 ;                  /* max($\mathbf{v}_d$.v) of all the directions, v, considered */
jump ← 0 ;                                  /* size of the jump which defines a probable path */
if open_paths.size() > 0 and probable_paths.size() > 0 then
    next_opening ← BestOpenPath(open_paths, $\mathbf{v}_d$, dot_max);
    next_opening_p ← BestProbablePath(probable_paths, $\mathbf{v}_d$, dot_max_p, jump);
    open_paths.clear();
    if dot_max_p > dot_max and jump >min_jump then next_opening ← next_opening_p;
else if open_path.size() > 0 then
    next_opening ← BestOpenPath(open_paths, $\mathbf{v}_d$),open_paths.clear();
else if probable_path.size() > 0 then
    next_opening ← BestProbablePath(probable_paths, $\mathbf{v}_d$),probable_paths.clear();
else return false
return true
```

Function FindVelocity

```
input  : Agents properties,next_opening,$\mathbf{v}_n$
output: Whether finding a velocity is successful, next moving direction: $\mathbf{v}_n$
half_planes.clear();
FindNaviPlanes(half_planes,next_opening);            /* Creates half-planes with the openings */
min_navi_planes ← half_planes.size();
TuneParameters(crowd_density);      /* Chooses the parameters for the ORCA planes creation */
FindORCAPlanes(half_planes,neighbours_info,$\mathbf{v}_c$,$\mathbf{r}$);          /* Creates an neighbour ORCA planes */
while not LPSolver(half_planes,$\mathbf{v}_n$) do     /* If no feasible solution found relax conditions */
    half_planes.pop_back
end
/* min_navi_planes is the minimum threshold to consider LPSolver() to have failed.
Currently allows momentary collisions with other agents and disallows collisions with the
environment                                                                              */
if half_planes.size()<min_navi_planes then return false;
return true
```



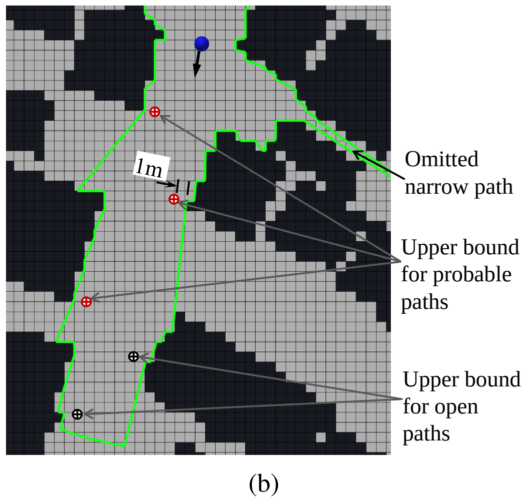


Figure 5.3: Vision features example

set of nodes obtained by eliminating nodes in-between visible nodes. In order to follow these nodes, which can be several tens to hundreds of meters apart, their desired direction is updated with the direction to the next node and the vision based navigation is used to navigate between the nodes defining the path. Since non-visitor agents know the path to the destination the size of the jumps to consider the probable path is smaller than for visitor agents.

5.2.2.2 Navigation for Visitor Agents - $g_v^{navigate}$

The visitor agents don't posses any information about their environment aside of what they perceive based on the information they receive from g^{see}; the sight distance may vary between 30m to 100m or more. This lack of information results in the inability to plan a path to a destination, or even being unaware of a concrete destination. In order to be able to reach a safe destination the visitor agents have to get information from non-visitors, mass media, or follow a group. As the currently considered evacuation scenario is a tsunami evacuation, the destination information could be deduced from visible high grounds or tall buildings. This is an ability any real person has. Once people start evacuating due to a tsunami warning, they should be able to identify a high ground and evacuate in that direction through the urban environment. To give the agents the ability to find a path through an unknown environment based on what they see, the Algorithm 1 is implemented. It is found that choosing the next opening as shown in Algorithm 1, allow the visitors to navigate through the environment and reach a safe destination.

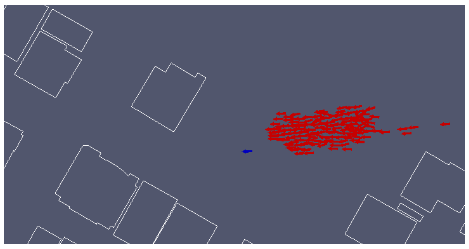


Figure 5.4: Agent following

5.2.2.3 Visitor agent evacuation strategy - $g^{find\_and\_follow}$

First explain why you need this function. Starts by visually recognizing someone to follow, from now on referred as a target. If a visitor doesn't have a target or the target is out of vision it tries to find a new target to follow. Only the agents moving away are considered as candidate targets and the one with the smallest speed difference is chosen as the target. If no suitable agent to be followed can be seen it will continue walking in the direction of the last seen target. If the target wasn't moving in the previous time step it stops following it. In order to avoid possible overcrowding, visitors extrapolate the target agent's movement and sets the extrapolated point as their temporary desired destination. Depending on the distance to the targets, visitors adjust speed to keep up with the target.

5.2.2.4 Interaction with the dynamically changing environment - $g^{is\_path\_blocked}$

Right after an earthquake disaster, the damages caused by the earthquake are not known to evacuees, unless they witness or encountered those while evacuating or receive information from other agents who experienced those. To model this, the grid environment is dynamically updated, according to the earthquake induced damages $\lambda^{earthquake}$ and/or inundation data $\lambda^{tsunami}$. Agents visually discover incongruences by comparing the available paths in their mental map and what they perceive with their vision, and update their mental maps based on these experiences. Figure 5.5 shows the initially planned path of a resident agent and the re-planned path after visually detecting that the previous path is blocked.

5.2.3 Path planning algorithms - $g^{path\_planning}$

In order support complex behavioral models several standard path planning algorithms are included: shortest path, closest destination and path to that, shortest path with way-points, k-mutually independent paths, etc. These are insufficient for the target applications, where people tend to choose safer paths according to time availability and perceived level of danger. Especially, in the case of earthquakes people tend to take paths with longer stretches of wider roads to lower the probability of encountering blocked roads, depending on the intensity of ground shaking they experienced. The graph provided in the Hybrid

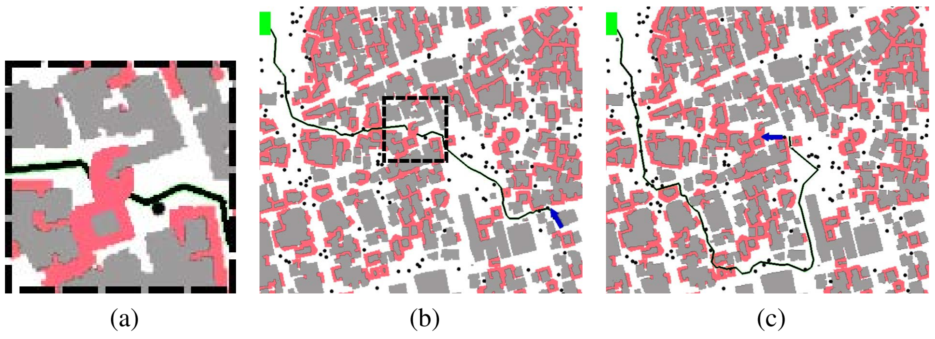


Figure 5.5: Blue arrow, observed agent. Left, blocked road zoom in. Center, initial path plan as the agent doesn't know the road is blocked. Right, as the agent gets close enough and visually identifies that the road is blocked, it re-plans its path.

environment allows performing path planning efficiently as it is a lighter representation of the environment. Furthermore this graph provides the required parallel enhancements to be easily shared and condensed among parallel processes.

5.2.3.1 Width/Number of agents preference

Standard Dijkstra algorithm allows the use of strong constrains such as minimum road width. But for the above mentioned purpose weaker constrains reflecting the preference for wider roads are needed.

Algorithm 5.2 shows the modified Dijkstra algorithm to find the destination which has the longest stretch of the path with preferred width and reachable within the given time/length constraints. The required inputs are: a known node in the topological graph nearest to the agent's current location, minimum preferred width, preferred width and level of preference. The preference is introduced as a bias in the way the graph edges lengths are perceived. Roads narrower than their minimum preference are unaffected, while roads wider than its preference will be perceived to be shorter in length. The reduction factor is interpolated for the roads in between (see Algorithm 5.2 lines 9-11). While exploring nodes of the graph, we keep track of actual distances, and a node is explored only if the actual cumulative distance is less than the maximum allowable for the agent (see algorithm 5.2 line 14). This maximum allowed distance is an agent's personal goal, currently calculated using the tsunami arrival time and the agent's average speed, which is an ideal case.

In the case of volunteers, the same algorithm using the number of agents as the bias parameter, instead of the width, is used to make them find paths giving priority to roads with more evacuees.

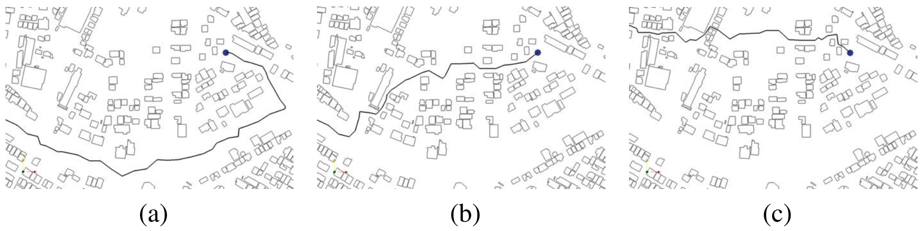


Figure 5.6: Effect of different levels of preference, evacuation area assumed to be on the left edge of the domain

5.2.3.2 Path finding with preference level example

Path planning using different levels of preference with the modified algorithm is ilustrated in Figure 5.6. The considered range of preference level factors is from $(0, 1]$. The lower the value the higher interest of the agent to traverse wide roads. The path with preference level 1 is equivalent to the shortest path found with the unmodified Dijkstra algorithm. The use of values above 1 would introduce aversion to the use of wider roads. This could potentially be used to command official agents and rescue services to search for lost visitors in narrow neighborhoods.

Algorithm 5.2 Modified Dijkstra algorithm

```
input  : Starting node $s$, preference factor $pF$, boolean
         functor IsADestination(), interpolation function
         In(), maximum allowed distance $l_{max}$, Positive
         weighted graph $G = (V, E, l, \boldsymbol{w})$ ($l$-length,$\boldsymbol{w}$-width,
         of the edges), $\boldsymbol{w_{min}}$, $\boldsymbol{w_{pref}}$
output: $path$ to destination and its $length$

1  for $\forall u \in V$ do pd($u$)= $\infty$; parent($u$)= $-1$;
   /* d-distance,pd-perceived distance, nd-new
   distance, npd-new perceived distance                */
2  pd($s$)= 0; d($u$)= 0;
3  $priority\_queue$.add($s$,pd($s$));
4  while not $priority\_queue$.empty() do
5      $u = priority\_queue$.top();
6      if IsADestination($u$) then break();
7      $priority\_queue$.pop();
8      for $\forall(u, v) \in E$ do
9          if $w(u,v) < w_{min}$ then $pl(u, v) = l(u, v)$;
10         else if $w(u,v) > w_{pref}$ then $pl(u, v) = pF * l(u, v)$;
11         else $pl(u, v)$ =In($pF, w_{min}, w_{pref}$)$*l(u, v)$;
12         npd($v$)=pd($u$)+$pl(u, v)$;
13         nd($v$)=d($u$)+$l(u, v)$;
14         if pd($v$) $>$ npd($v$) and npd($v$) $< l_{max}$ then
15             d($v$)=nd($v$);
16             pd($v$)=npd($v$);
17             parent($v$)=$u$;
18             $priority\_queue$.add($v$,pd($v$));
19 if IsADestination($u$) then // Extracts the path
20     $length$ = d($u$);$path$.pushBack($u$);
21     while $u \neq s$ do $u$ = parent($u$); $path$.pushBack($u$);
22 return $path$ , $length$
```

Chapter 6

Demonstrative applications

In order to demonstrate the usage of the evacuation simulation tool and highlight the need of detailed modeling, several scenarios testing different evacuation strategies and mitigation measures involving mixed mode (car and pedestrian) evacuation are evaluated. The aim of these scenarios is to find out the effect of the usage of cars in the evacuation and find means to improve the evacuation throughput with their usage.

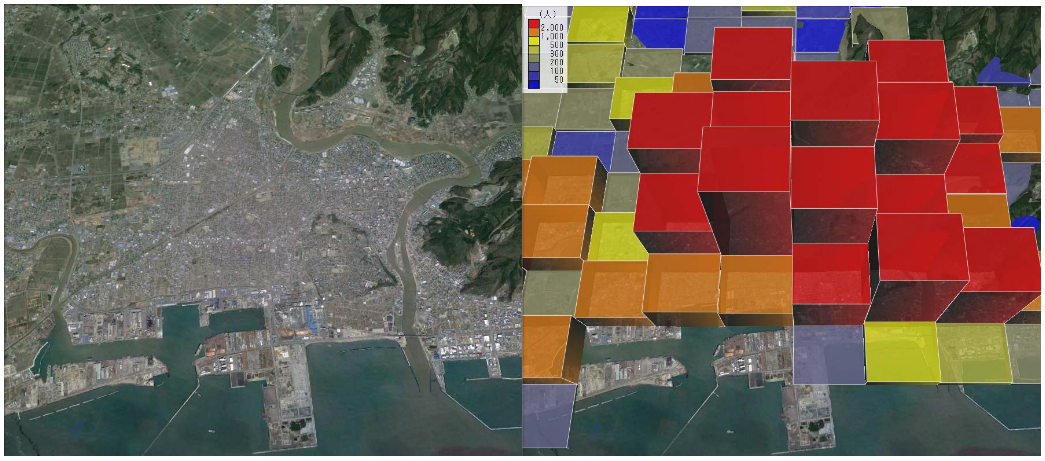
It should be emphasized that these simulations are solely conducted to demonstrate the potentials of the developed evacuation simulator, and the results must not be used for any practicle purposes.

6.1 Setting

The setting for these demonstrative examples is Ishinomaki city, Miyagi prefecture Japan, see fig. 6.1a. This is a densely populated urban area, figure 6.1b shows the population distribution in $1km \times 1km$ blocks. From this area a region of $6km\times$ 6.5 km is chosen for the analysis, see Fig. 6.2. This region has historically suffered from tsunami catastrophes, see fig. 6.3a, and its flat topography as seen in figure 6.3b increases the vulnerability to tsunami inundation. The synthetic population is initialized as seen in table 6.1 and their starting locations are restricted to the most populated region. 40,000 evacuees are considered. The tsunami arrival time is assumed to be 40 minutes as observed during the 2011 Great East Japan Earthquake and Tsunami. The pre-evacuation time, the time it takes for an evacuee to start evacuating after the first earthquake shock, is assumed to follow a normal distribution. The mean and standard deviation for this pre-evacuation time for pedestrians is obtained from literature[40]. Like most of the small coastal towns in Japan, a significant fraction of the population in this area is elderly. Hence, the population is subdivided in two groups each constituting half of the population; pedestrians fast representing young people, and pedestrians slow representing elder people. The speed of each of each group is assumed to follow a normal distribution. Each vehicle is considered to carry 2 evacuees, representing a pessimistic usage of vehicles.

	Pedestrian fast	Pedestrians slow	Cars
Speed mean (m/s)	1.5	1.0	9
Speed S.D. (m/s)	0.4	0.4	2
Speed bound low(m/s)	0.7	0.7	4
Speed bound high(m/s)	2.5	2.5	13
Pre-evacuation time mean (min)	15	15	variable
Pre-evacuation time S.D. (min)	5	5	5
Percentage (%)	50	50	variable

Table 6.1: Synthetic population parameters



(a) (b)

Figure 6.1: Interest area from Google Earth (left), population distribution [38] (right)

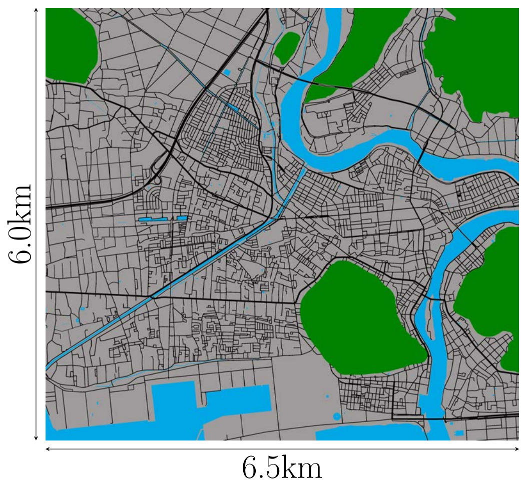
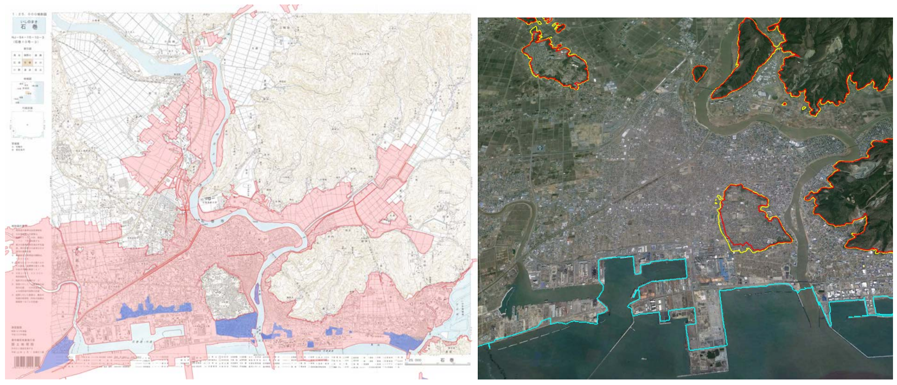


Figure 6.2: Evacuation area model



(a) Inundation level (pink) and washed out infrastructure (blue) during 2011 Great East Japan Earthquake and Tsunami [39]

(b) Contour line map[38]

Figure 6.3: Region vulnerability

6.2 Preliminary analysis

This section presents the assessment of the robustness of the results and the analysis of the required amount of resources needed to provide a good characterization of the distribution of these results. The central an innovative contrubution of this work is the inclusion of micro-scale interactions among evacuees, especially car-pedestrian interactions. The effect of these micro-scale interactions are quantified and the contribution to the evacuation throughput is evaluated. As mentioned in section 3.1.1.1 the behavior of car agents at intersections is conservative, a single car can use the intersection at a given time, disregarding the capacity of an intersection. The effect of this conservative intersection model is quantified and its contribution to the evacuation throughput is evaluated. Additionally, in order to evaluate the effect of having a different percentage of evacuees use cars in different parts of the domain based on their vulnerabilities, a regioning system for the demonstrative application is considered. For the preliminary analysis it is assumed that 25% of the population use cars for evacuation.

6.2.1 Monte Carlo Simulations

The instantiation of the synthetic population involves the initialization of random variables according to a given probability density function, e.g speed, pre-evacuation time, position in the domain. This allows to take into consideration the uncertainties involved in real life. In these demonstrative examples, the evacuees are assumed to be uniformly distributed across the domain. The random assignment of agent properties and location constrains our model to be assessed in a stochastic manner. Different draws from the probability distribution functions to initialize the random variables produce different outcomes in the simulation. However by relying on the law of large numbers, with a sufficient number of results, a stable average outcome can be provided. Furthermore, the variability of this average outcome can be evaluated once the distribution of the results are known.

In this section the robustness is evaluated through the use of Monte-Carlo simulations. Figure 6.4a, shows the convergence of the standard deviation of the results with the number of draws/simulations. As it can be seen, after about 600 simulations the standard deviation has already converged. Figure 6.4b shows the mean and the standard deviation of the results of the cumulative number of agents evacuated with 1000 Monte-Carlo simulations. The zoomed in box on the upper right corner shows a standard deviation of 0.42% in the throughput after 40mins, and in the lower right corner 0.29% after 25mins. Additionally, Fig. 6.4c shows the convergence of more sensitive measure, number of agents evacuated at each 10 sec interval (i.e. loosely speaking the derivative of the Fig. 6.4b). As is seen, the graphs for 600 simulations is almost identical to those of 1000 simulations with neglegible difference, indicating that both the mean and the standard deviations of this sensitive measure also converges after 600 simulations. It is important to note that this evaluation of the robustness provides estimates only for a specific scenario, with a few number of random variables. Ideally, evacuation process involve more random variables and the robustness should be evaluated for every scenario. However, for these demonstrative applications it is considered by the author enough to be known that the single results presented are drawn from distributions similar to the ones presented in this section which show low variability.

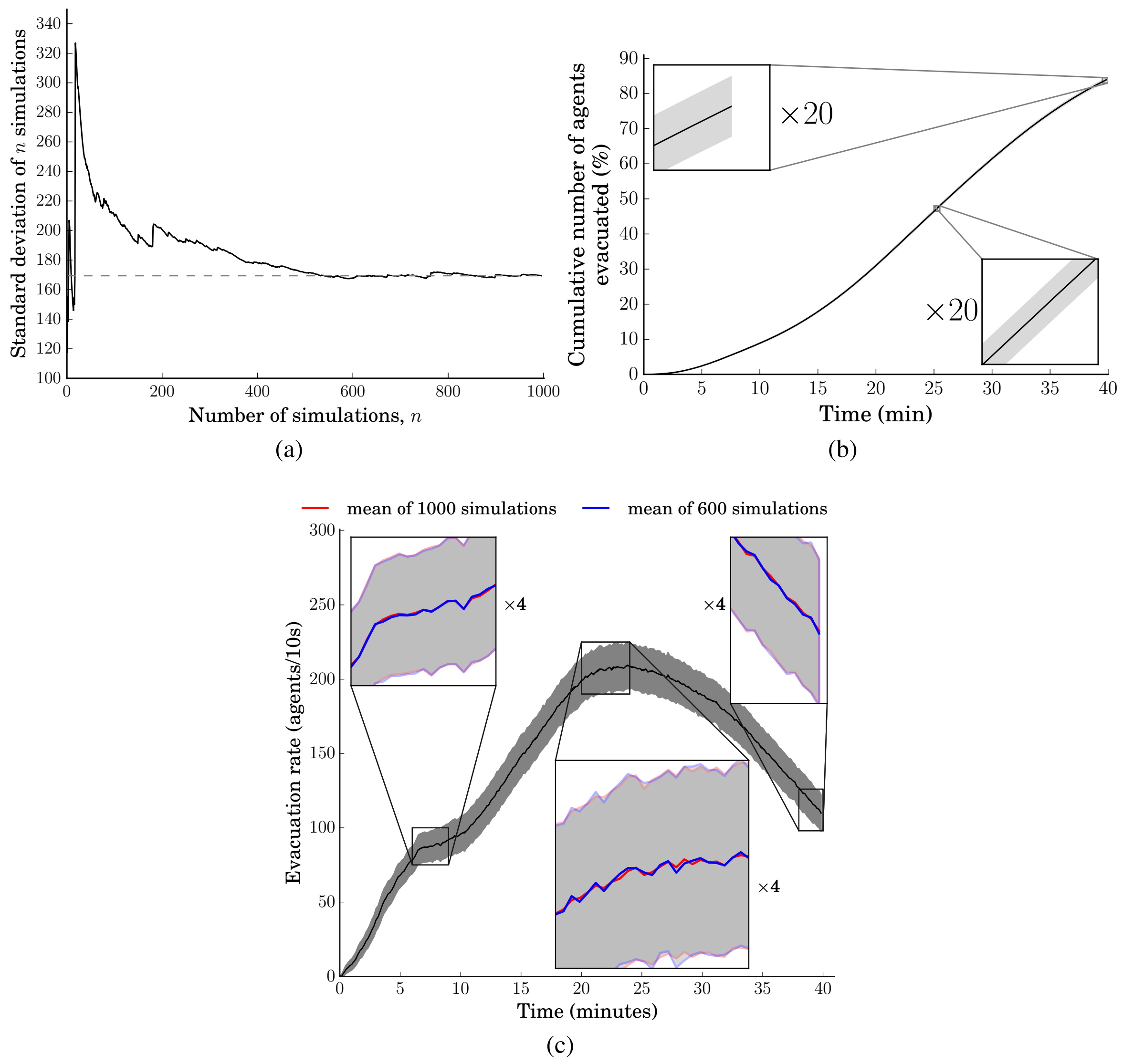


Figure 6.4: Monte-Carlo simulations robustness results

6.2.2 Effect of car-pedestrian interaction

The main innovative point of this work is the inclusion of micro-scale interactions among evacuees. In this section, the effect of car-pedestrian interaction on the evacuation throughput is evaluated. It is important to note the effect of this interaction model is expected to vary depending on the configuration of the evacuation domain, e.g. number of evacuees (density), width of roads and their distribution, physical dimensions of the agents and cars. Figure 4.3.3, compares graphs of cumulative number evacuated vs. time with car-pedestrian interactions enabled and disabled. As seen, the car-pedestrain interaction lowers the total throughput about 5%. difference in total throughput can be observed.

Note that the above evaluation does not correctly capture the effect of the car-pedestrian interaction during an emergency evacaution; the observations used for calibration are from ordinary pedestrains during a festival occation. However, it is reasonable to expect that the car-pedestrian interactions during an emergency evacaution has larger negative impact compared to the scenario considered here.

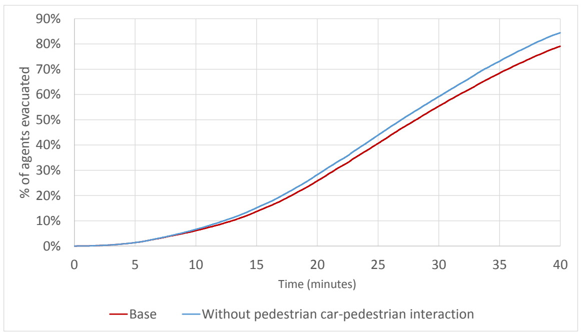


Figure 6.5: Effect of car-pedestrian interaction, red, interactions enabled (base), blue car-ped interaction disabled

6.2.3 Effect of car-car interaction at intersection

The current model of intersections allows a single car to use an intersection at a given time, disregarding the actual intersection's actual capacity. The larger the number of roads forming the junction and the number of lanes in those roads, the more the severity of this simple intersection model. The current simple interaction model provide an pesimistic estimation, especially when there is less traffic on the roads. In order to contrast this intersection model, a free flow intersection model is implemented by disabling the car-car interaction at intersections. In the case of the free flow intersection, any car is allowed to use the intersection disregarding the number of cars already using the intersection. This presents an optimistic use of the intersections which is impossible in reality. Figure 6.6, shows a comparison between the overly optimistic scenario with the free flow model and the single car model. A difference of about 4% after 40mins of evacuation can be observed. Intuitively, the results of a better model of intersection would lie in between these optimistic and pessimistic estimation.

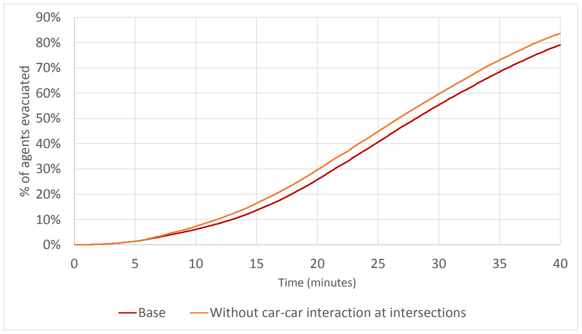


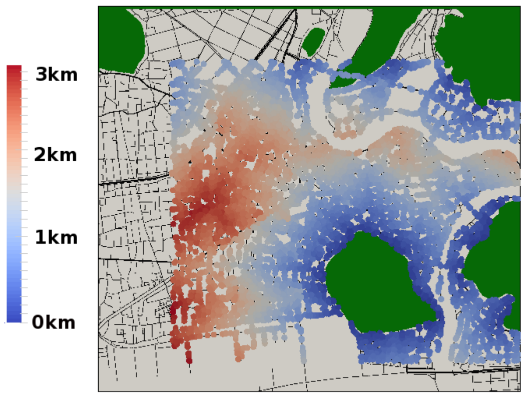
Figure 6.6: Effect of car-car interaction at intersection

6.2.4 Sub-regioning

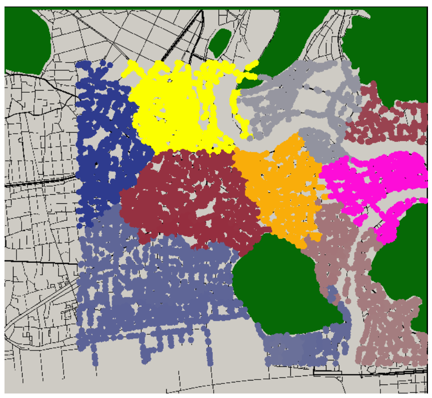
In order to evaluate the effect of different car usage in different sub-regions of the domain, first a sub-regioning system is to be defined. In practice the administrative divisions of the region should be evaluated to be used as a sub-regioning scheme. In the case of this demonstrative application a sub-regioning system based on the independence of flows and the vulnerability of each region is defined. This process starts by finding out the closest evacuation area for uniformly distributed locations in the domain and the shortest path from each location to its corresponding closest evacuation area. From this analysis the shortest distance an evacuee, starting at these locations, would have to traverse to reach the closest evacuation point is determined, see fig 6.7a. From this analysis the most vulnerable areas are identified, areas from where pedestrians would have to walk longer to reach an evacuation area are considered to have a higher vulnerability. Additionally from this analysis sub-regions of the domain with similar final destination are grouped as seen in fig. 6.7b. These areas represent regions of the domain, that agents starting to evacuate on those regions would evacuate almost independently from evacuees starting in other regions. With this information groups with high and low vulnerability are created. These groups are created as their different level of vulnerability and conditions require different strategies to improve their evacuation. The grouping threshold is set arbitrarily, these groups could be created with a finer grain if need. Ideally each group should be subdivided into vulnerable and less vulnerable regions, but in order to keep the number of strategical sub-regions low several sub-regions are merged together obtaining the final sub-regioning scheme as seen in fig 6.7c, where sub-regions 1,3 and 5 represent the most vulnerable areas and regions 2,4,6,7 represent the areas with low vulnerability.

Only considering the strategies of individually delaying or postponing the evacuation of different regions, the number of different schedules is given by the following equation.

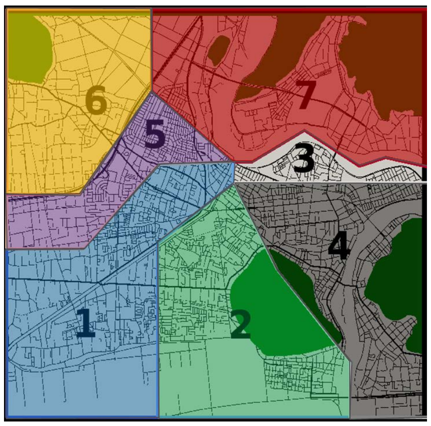
$$number\_of\_shedules(n) = n!(1-n) + \sum_{i=0}^{n-1} (n-i)! \binom{n}{i+1}$$



(a) Distance distribution



(b) Closest evacuation area grouping



(c) Regioning

Figure 6.7: Creation of a synthetic sub-regioning

From here it can be seen that even for a small number of sub-regions exhaustively testing every schedule is an impossible task to perform. Because of this, the strategies most likely to produce better results are handpicked to be tested.

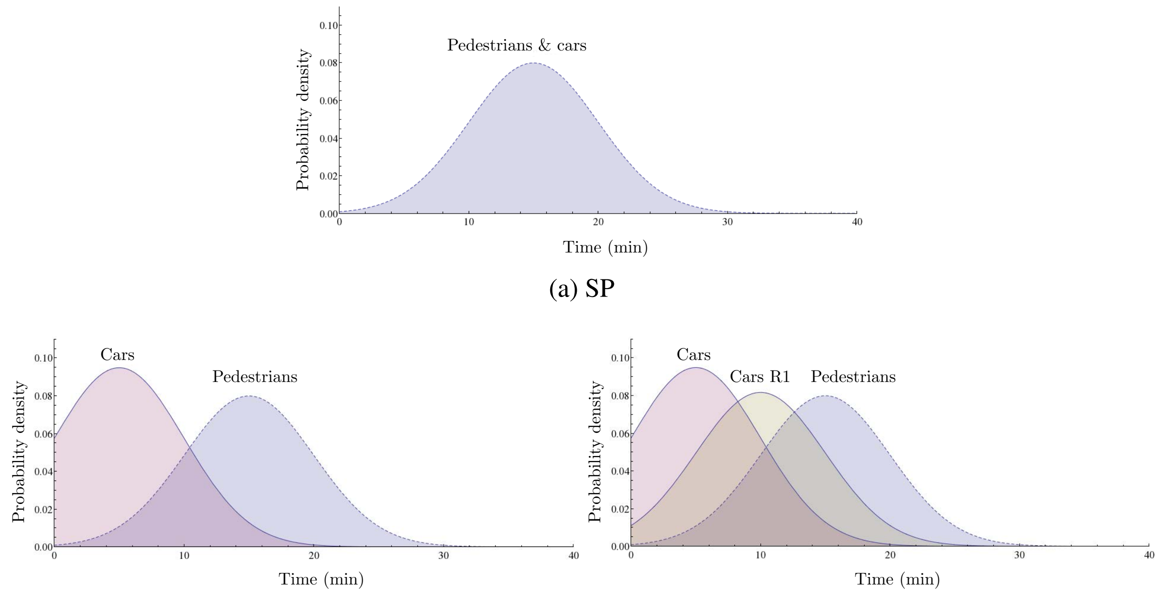
6.3 Scenarios

Different scenarios exploring possible strategies to take advantage of the evacuation using cars are analyzed in this section. The scenarios are built up progressively as pitfalls in the evacuation are found and different strategies are envisioned, in other words these scenarios are built up in an iterative way trying to improve the evacuation throughput at each iteration.

6.3.1 Scheduling

The first set of scenarios analyze the effect of 3 basic schedules, see Fig. 6.8. These schedules represent the basic scenario where: the pre-evacuation time distribution for cars and pedestrians is the same "same as pedestrians" (SP), Fig. 6.8a; evacuees using cars are persuaded to start to evacuate earlier with a mean of 5 min "cars first" (CF); evacuees using cars are persuaded to start to evacuate earlier with a mean of 10 min for region 1 and 5 min for the rest of the regions, "delay sub-region #1" (DF), Fig. 6.8c.

The schedules persuading the early use of cars try to solve one of the seriousproblems in the tsunami evacuation; the delay in the strating of evacuation which causes individuals to be unable to evacuate. In this case the usage of cars would be used as an incentive for early evacuation. These schedules are considered as the best candidates to give a better insight to the evacuation process in the interest area.



(a) SP

(b) CF

(c) DF

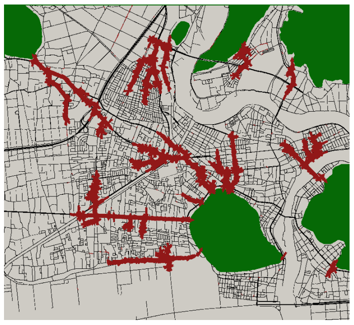
Figure 6.8: Basic schedules

6.3.2 Indiscriminate use of cars

The effect of the schedules is combined with the effect of different percentages of evacuees using cars. The percentages of evacuees using cars tested are 0% (no evacuees using cars), 25%, 50% and 75%. This scenario evaluates the effect of the indiscriminate use of cars where no restriction is imposed in who is allowed to use cars.

Figure 6.10 shows that, although the evacuation using cars is thought to be faster, this improvement or boost in the evacuation is only appear during the early stages. As the percentage of car usage increases the roads get saturated earlier and the benefits of the usage of cars are lost afterwards. Additionally, this initial boost in the evacuation throughput is improved by the schedules persuading cars to evacuate first.

Moreover, these results show that the indiscriminate use of cars could potentially be harmful for the overall throughput of the evacuation, as none of the schedules produce a total number of evacuees reaching a safe area higher than the base pedestrian only evacuation. In the worst case scenario, car usage of 75%, the evacuation throughput after 40mins dropped by 17%. Through detailed analysis of the evacuation results, see Fig. 6.9b, it is observed that the emergence of long cars queues reduces the effectiveness of scheduling. Especially "delay sub-region #1" and "cars first" schedule's results become indistinguishable. It can be observed that delaying any of the vulnerable regions doesn't separate the flows (the desired effect), instead it only postpones the entrance of the cars to the queue.


(a)

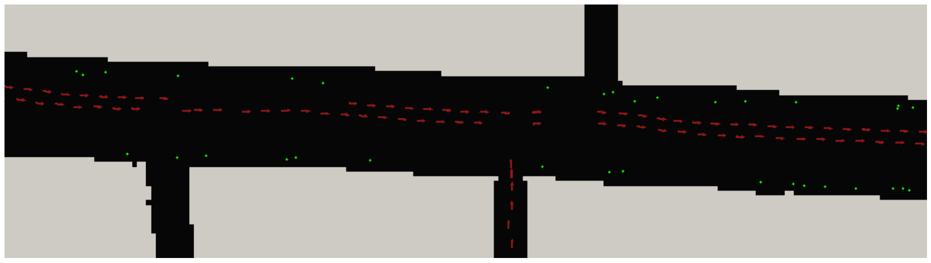
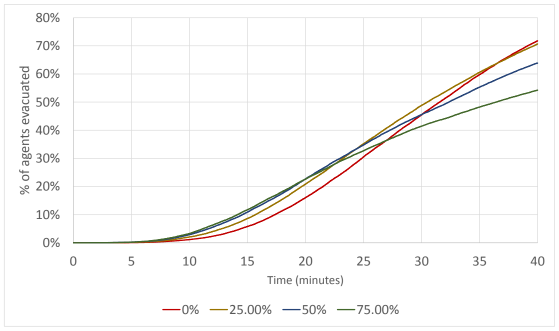
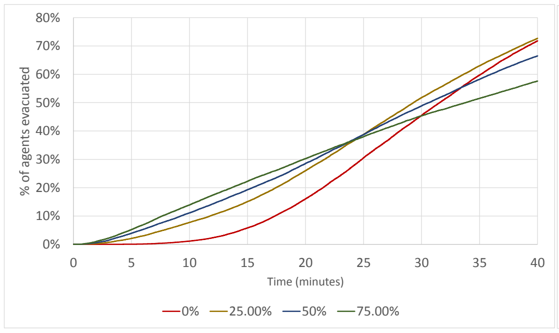

(b) Vehicles in red, pedestrians in green, road in black

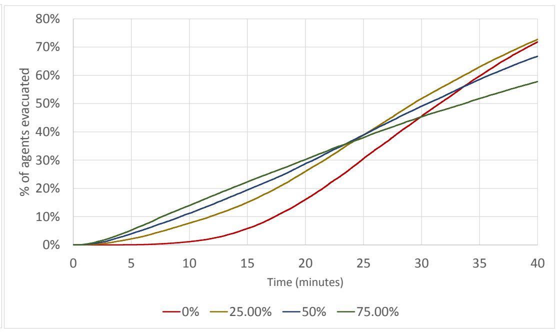
Figure 6.9: Queue emergence, macroscopic view (a), microscopic view (b)

It is important to note that the evacuation using cars even-though doesn't produce an overall throughput higher than the one of the base pedestrians only scenario the initial boost in the evacuation could prove itself useful in scenarios with shorter tsunami arrival time. This fact could be used in practice as a safeguard. Table 6.2 shows the best and worst throughput of every schedule.



(a)





(b) (c)

Figure 6.10: Indiscriminate use of cars results

6.3.3 People in need restriction

In the batch of scenarios presented in the previous section, the usage of cars is indiscriminate, any evacuee can use a car disregarding their vulnerability. In the set of scenarios presented in this section the results of previous section are reproduced but restricting the car usage to people in need. People in need are defined as the evacuees having the slowest evacuation speed in the synthetic population and the usage of cars is restricted for those most in need. Table 6.3 shows the summary of results.

The results obtained by restricting the use of cars for people in need can be seen in fig. 6.11. It can be seen that by applying this restriction and improvement in the total evacuation throughput of about 7% over the base scenario can be achieved. Additionally higher percentages up to 50% of the population can use cars without a significant impact in the total evacuation throughput. Moreover, it can be observed that the queue emergence remains as a problem as the changes in these scenarios do not provide any way of avoiding it.

	Best		Worst	
Schedule	Car use (%)	Total (%)	Car use (%)	Total (%)
Same as pedestrians	0	71.80	75	54.25
Cars first	25	71.82	75	57.64
Delay #1	25	71.82	75	57.79

Table 6.2: Summary of best and worst scenarios, basic scheduling

	Best		Worst	
Schedule	Car use (%)	Total (%)	Car use (%)	Total (%)
Same as pedestrians	25	77.06	75	59.98
Cars first	25	79.25	75	63.48
Delay #1	25	79.23	75	63.68

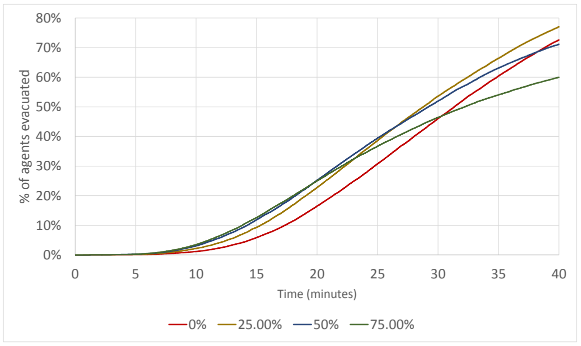
Table 6.3: Summary of best and worst scenarios, people in need restriction

6.3.4 Widening of roads

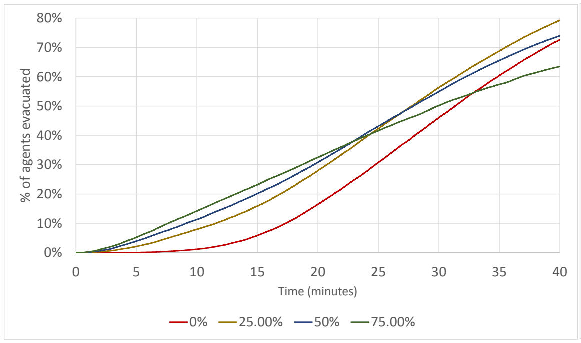
In order to quantify the effect of widening the road network to increase the throughput of the evacuation, scenarios increasing the width of the whole road network by $1m$ on each side are tested. Although this scenario is not feasible as the cost of increasing the width of the road network would be too high, it showcases the ability of the simulator to quantify such scenarios and highlights the need of detailed information of the state of the simulation. Such information would be needed to focus the available resources. The scenario used in the previous section is used as a base: 3 schedules, 4 different percentages of car usages and the restriction for people in need. Additionally a domain with all the roads widened by $1m$ on each side is used. Table 6.4 shows a summary of the best and worst results of these scenarios.

Figure 6.12 shows the time history of the evacuation. It can be seen that the improvements provided by this scenario are marginal, in the order of 2%. The cost of improving the whole road network and the marginal improvement gained by doing so makes this approach impractical. As it will be shown in the following sections, the evacuation simulator provides way of identifying road network segments based on their usage, speed, or other characteristics. This would allow evacuation planners and policy makers to decide where to focus the resources in order to maximize the benefit from them.

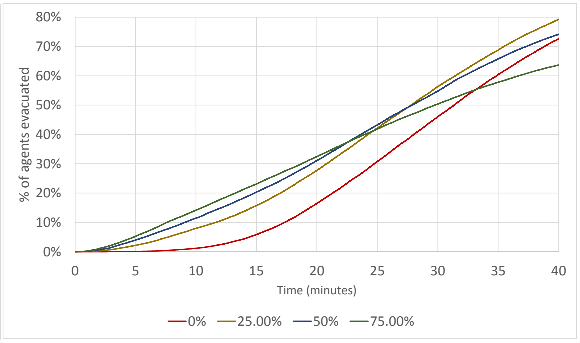
Surely, the widening of all roads by $1m$ is not a practical solution, better approaches like widening only the roads which suffer from traffic congestion or make new roads to divert the traffic should have noticeable improvement. The tools to discover the locations of congestion or problematic roads are presented in section 6.3.6.



(a)



(b)

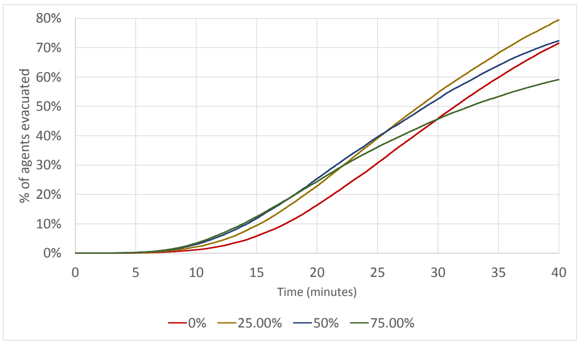


(c)

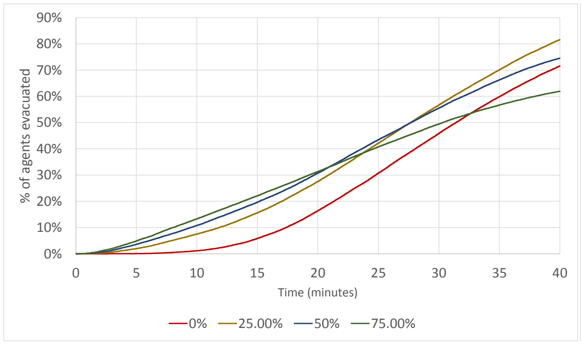
Figure 6.11: People in need restriction results

6.3.5 Fine tuning the percentage of cars

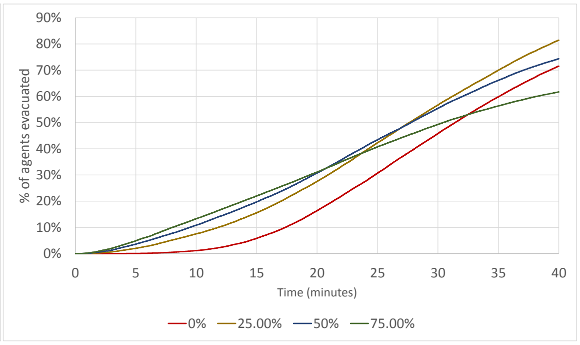
In section 6.3.3, the evaluation of the usage of cars by people in need is performed. In this section the vulnerability of each region is introduced as an additional factor to consider. The sub-regioning system introduced 6.2.4 is used for these scenarios. Regions closer to evacuation areas are considered to be less vulnerable thus requiring less people in need to take advantage on the use of cars, while regions further away from the evacuation point would require higher percentages as more pedestrians in this region wouldn't be able to reach otherwise. Under those assumptions scenarios aiming to evaluate the effect of different percentages of cars in the evacuation are evaluated, additionally to the previously defined schedules. As the number of possible combinations is too big to be evaluated exhaustively, areas are separated in vulnerable (areas 1, 3,5) and less vulnerable (areas 2,4,6,7), and different percentages of cars are evaluated in each of them. For vulnerable areas 5% increments are considered starting with 20% upto 40% of car usage, and for less vulnerable region 10% increments are considered starting from 15% up to 35%. Table 6.5, provides a summary of the car usage percentage per scenario with the best performing scenario for each schedule highlighted in bold.



(a)



(b)



(c)

Figure 6.12: Widening of roads results

Figure 6.13 provides a comparison among the best performing scenarios under the three different schedules. The total throughput and the best performing scenario are as follows for each schedule; 77.20% in scenario #2 under "same as pedestrians" schedule, 79.21% in scenario #2 under the "cars first" schedule and 78.95% in scenario #3 under the "delay sub-region #1" scenario. See table 6.5 for details. The differences between the best result from schedule cars first and delay first are negligible; in depth observation of the simulation results still shows the formation of queues as described in the previous section which disables the ability to capitalize the difference in schedule.

Figure 6.14a, shows the results of all 15 scenarios under the same as pedestrians schedule as this schedule provided the best total throughput. The coloring in this region is set grouping the changes in the least vulnerable regions. It can be observed that the changes in the percentages on the least vulnerable regions separate the initial segments of the graphs being this difference most clear at around 15minutes of the evacuation start. It can be seen that greater percentages of cars in less vulnerable (closer) regions is what provides these initial boots in the evacuation throughput, but it also degrades the total throughput as cars in the more vulnerable regions (farther) find the roads already saturated when trying to

	Best		Worst	
Schedule	Car use (%)	Total (%)	Car use (%)	Total (%)
Same as pedestrians	25	79.49	75	59.14
Cars first	25	81.56	75	61.93
Delay #1	25	81.52	75	61.17

Table 6.4: Summary of best and worst scenarios, people in need restriction

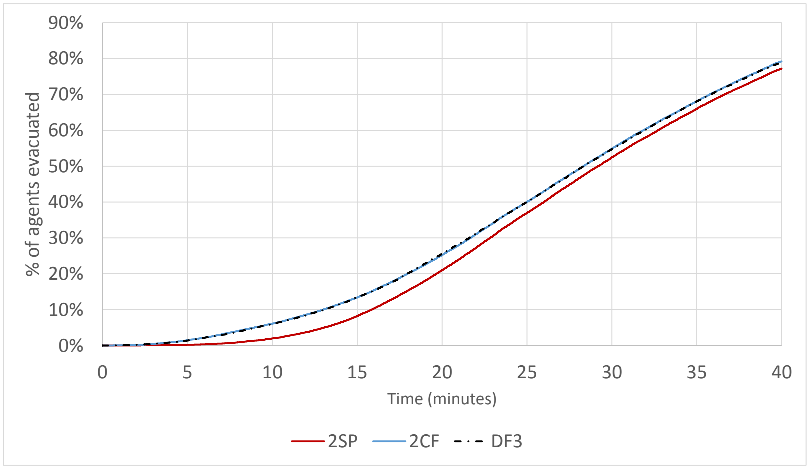


Figure 6.13: Summary extracting the best performing scenarios from each schedule (S - same pedestrians, DF - delay first, CF - cars first

reach the evacuation area. Figure 6.14b extracts from the 15 scenarios the best and worse scenarios using the total throughput after 40mins as a discerning variable. The difference in the total throughput is around 2%, at 40 minutes.

Detailed observation of the results of the best performing schedule, cars first, shows that the pedestrian throughput is improved by allowing people in need (slowest agents) to use vehicles, see figures 6.15a, 6.16a, this is because the reduction of slow evacuees on foot allow crowds to move faster as a whole. In contrast, cars throughput is reduced with the addition of more evacuees using cars, as seen in figures 6.15b, 6.16b. This phenomena is expected as traffic queue formation becomes critical with the addition of vehicles.

The coloring scheme used in 6.15a and 6.15b, varies the color by the changes of percentages of car usage in the vulnerable regions and presents the results as percentages of the same agent type. As the percentage of cars in the vulnerable regions increase the percentage of pedestrians in relation to the number of pedestrians increases and the percentage of cars in relation to the number of cars decreases. The best performing scenario is presented in dashed black.

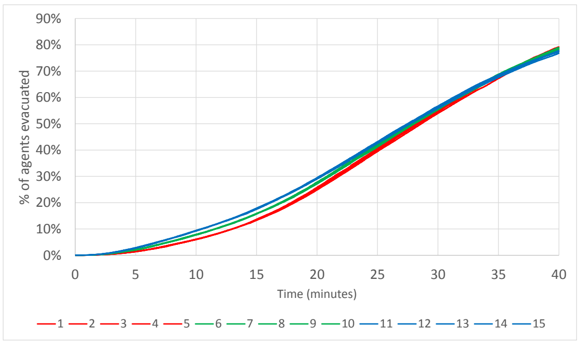
		Region (% of cars)							Total throughput (%)		
		1	2	3	4	5	6	7	SP	CF	DF
Simulation number	1	20	15	20	15	20	15	15	76.82	78.02	78.58
	2	25	15	25	15	25	15	15	**77.20**	**79.21**	78.91
	3	30	15	30	15	30	15	15	76.70	78.54	**78.95**
	4	35	15	35	15	35	15	15	76.78	78.63	78.72
	5	40	15	40	15	40	15	15	76.78	79.09	78.71
	6	20	25	20	25	20	25	25	76.27	78.26	78.28
	7	25	25	25	25	25	25	25	76.53	78.96	78.92
	8	30	25	30	25	30	25	25	76.64	78.69	78.81
	9	35	25	35	25	35	25	25	76.40	78.12	78.12
	10	40	35	40	35	40	35	35	75.92	78.37	78.23
	11	20	35	20	35	20	35	35	75.65	77.77	77.62
	12	25	35	25	35	25	35	35	74.42	77.50	77.33
	13	30	35	30	35	30	35	35	75.11	76.79	77.25
	14	35	35	35	35	35	35	35	74.94	77.44	77.35
	15	40	35	40	35	40	35	35	75.26	77.39	77.34

Table 6.5: Scenario number, percentage of pedestrians using cars per area, and total throughput after 40mins

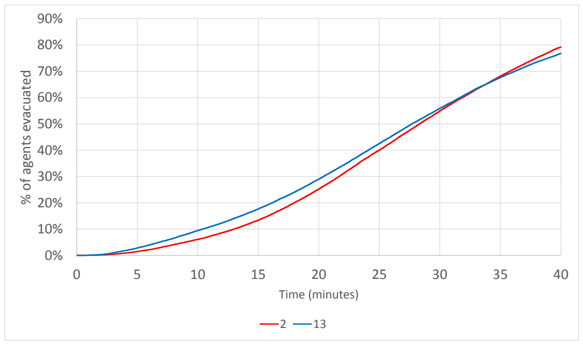
On the other hand figures 6.16a and 6.16b use the coloring scheme to highlight the changes in percentages in the less vulnerable regions. It should be noted that the best performing scenario plotted in black with dashes coincides with the best performing scenario for the pedestrians, showing that although this result is clearly sub optimal from the perspective of the vehicles it produces the best overall throughput.

6.3.6 Detailed data output

As seen in section 6.3.4, there is a need to identify critical road network segments in order to be able to prioritize and focus the available resources. The advantages of the use of agents in the modeling of the evacuation phenomena is that details on the behavior, thought process, interaction, etc. can be introduced, these details do not serve only to provide better control over the simulation and allow the simulation of more complex phenomena, but additionally allows the production of fine grained data. This fine grained data needs to be condensed and made available for its effective usage for evacuation planers and policy makers. The current process for performing this task can be found in B.1. The evaluation of the time history evolution of the evacuation can be assessed as in Fig. 6.17. Figure 6.18 shows examples condensing the average speed of the cars and the pedestrians on the road network. This information can be used to plan effective strategies and measures in the



(a) Results of all 15 scenarios under the cars first schedule



(b) Best and worst case from cars first schedule

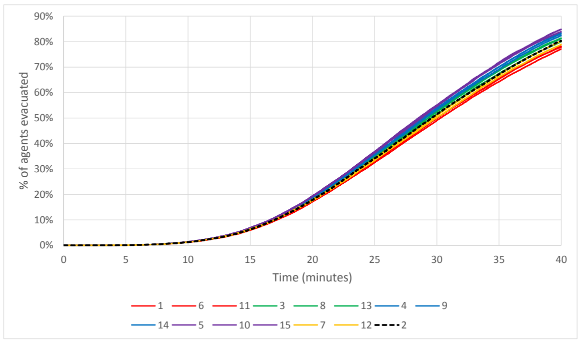
Figure 6.14: Finer control results

identified bottlenecks. With such information highly used roads where the average speed is low could be prioritized and different measures could be tested through the use of this simulator

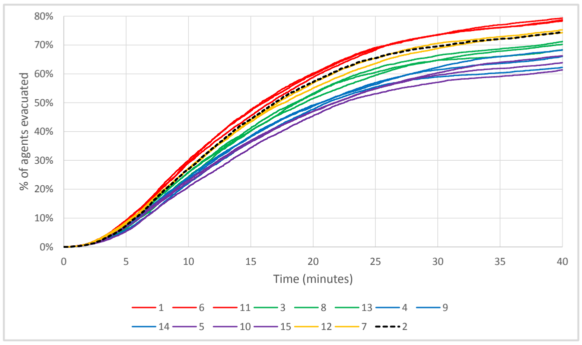
Additionally to the information condensed to the graphs other relevant information can be observed through the coloring of the agents. Figure 6.19 shows 3 snapshots of the evacuation positioning circles in the location of the agents exaggerating the circles sizes to make them visible at large scale and using their coloring to output the agents initial region. This kind of fine grained information provides additional means of identifying problems and insight to information needed for the planning answering with a quick glance questions such as, "Where did most of the people who were not able to evacuate come from? What evacuation mode were they using?"

6.3.7 Remarks

It is observed that the indiscriminate use of cars significantly hinders evacuation process. Restriction on the usage of cars to people in need is shown to provide improvements in

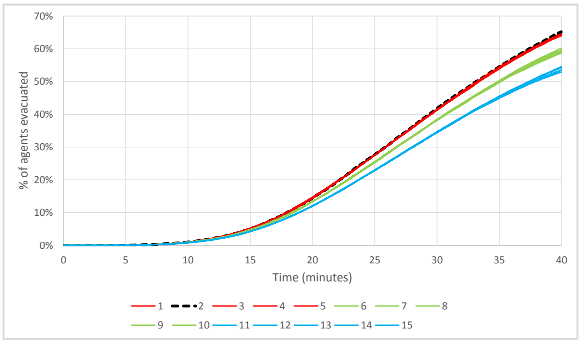


(a) number of pedestrians evacuated in relation to the number of pedestrians in the simulation

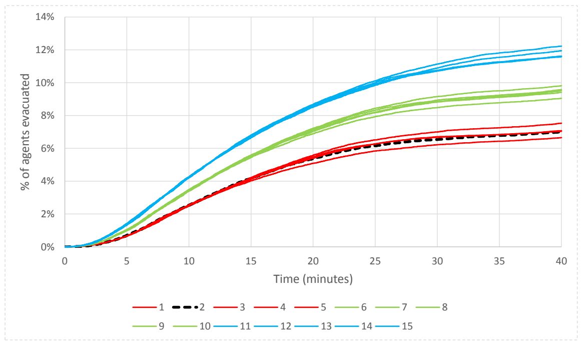


(b) number of cars evacuated in relation to the number of cars in the simulation

Figure 6.15: Effect of different percentages of cars per region for cars first schedule



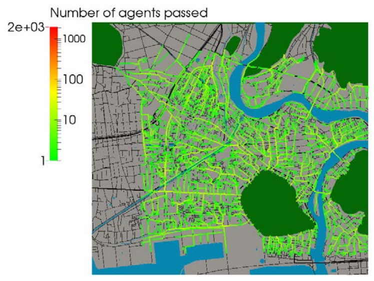
(a) number of pedestrians evacuated in relation to the total number of evacuees in the simulation



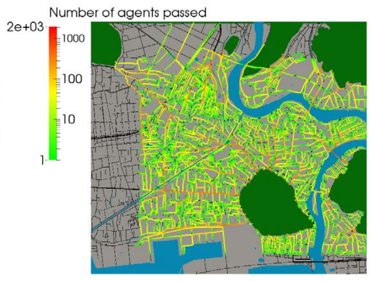
(b) number of cars evacuated in relation to the total number of evacuees in the simulation

Figure 6.16: Effect of different percentages of cars per region for cars first schedule (total)

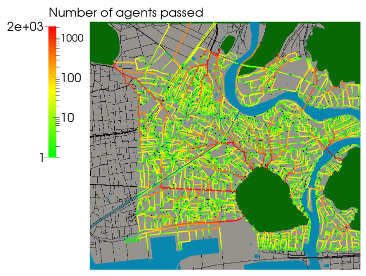
the evacuation throughput, as seen in section 6.3.3. Detailed modeling of the environment, agents and interactions and the production of fine grained data provides insights to the evacuation process necessary for effective planning. There is the possibility of finding better percentages of cars usage through the use of the automatic calibration tool and the treatment of the problem as a combinatorial optimization. Different combinations of schedules, usage of cars and other strategies can be evaluated in an automatic manner with this approach. This process could be further enhanced by improving the parallel performance of the simulator, additionally simplifying the testing and development process in large scenarios.



(a)

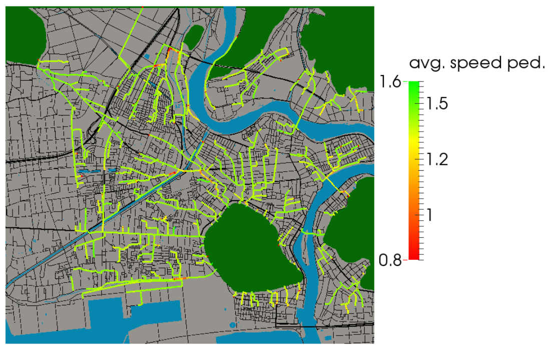


(b)

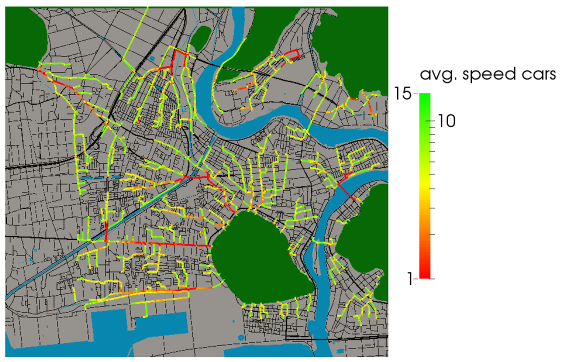


(c)

Figure 6.17: Road usage (10, 20, 39 mins)

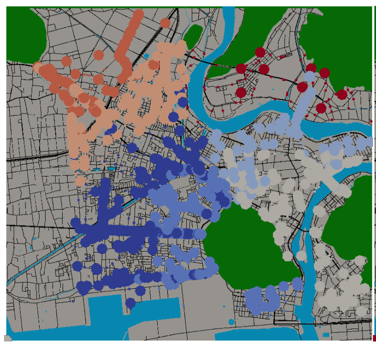


(a)

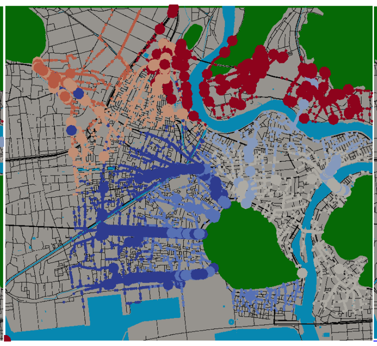


(b)

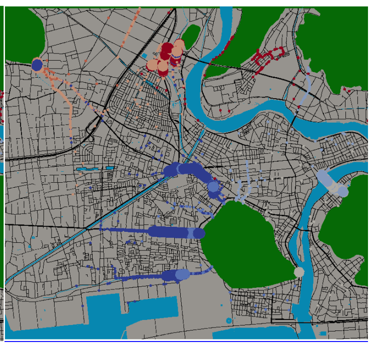
Figure 6.18: Average road speed after 40min of evacuation, pedestrians left, cars right in roads used by more than 50 evacuees



(a)



(b)



(c)

Figure 6.19: Time evolution of the evacuation state, with coloring by region (5,20,39 min)

Chapter 7

Concluding Remarks

Enhancements of an evacuation simulation software to enable evaluating mixed mode evacuations considering cars and pedestrians is presented. Micro scale interactions between evacuees using different modes of evacuation are introduced along with an environment capable of supporting these interactions. A mathematical framework is introduced to provide enough expressiveness to explain the different parts of the simulator along with the enhancements required to provide mixed mode evacuation evaluation capabilities to the software. Collision avoidance is used as the base of the transnational interactions among evacuees and the environment. Automated calibrating functionality is introduced so that the agents' parameters can be tuned to reproduced given observations of evacuees. The three translational interactions among the two groups of agents, representing pedestrians and cars, are tuned and validated based on observations.

To demonstrate potential usage of the enhanced evacuation simulator, mixed mode evacuation scenarios are considered in an urban coastal city in Japan. The effect of the car-pedestrian interaction and the car-car interactions at intersections are quantified in this scenario to identify their contribution to the overall results; about 5% from car-pedestrian interaction, and about 4% from the simple model of intersection. The robustness and distribution of the results for one scenario are evaluated through Monte-Carlo simulations. For the considered scenarios, it is estimated that 600 simulations are required to provide robustness to the results. Single simulation results of various scenarios are presented to highlight the potential of the system. Surely, Monte-Carlo simulations should be conducted, if the simulations are application oriented. Although the single simulation results are not completely conclusive as they lack robustness, these provide a good heuristic for the search of adequate strategies. Once a compelling strategy is found its robustness and the robustness of other competing strategies should be evaluated to provide a fair comparison among them. The base results of the present simulations, constrained to the assumptions of our model, are:

- The indiscriminate use of cars could have a dramatic negative impact on the total evacuation throughput, from the scenarios tested a car usage of 75% produces a drop of 17% in the total throughput (after 40mins, tsunami arrival time) of the evacuation.
- Any percentage of cars provides an initial boost in the evacuation throughput, which lasts for a short duration at the beginning and goes down with the increasing percentage of cars.
- Restriction of car usage to those in need provides the best throughput (after 40mins tsunami arrival time). When cars are restricted to people in need, 25% of the population could be allowed to use cars and the total number evacuated is increased around 5%, comapred to the case without car usage.
- It is shown that the usage of cars as a measure to persuade early evacuation, although provides marginal improvements in the total throughput, increases the initial evacuation throughput considerably; this is especially important in evacuation with earlier tsunami arrival time.
- The possibility to fine-tune the percentage of cars based in a domain regioning scheme is demonstrated; these fine tuning approach would require results with more robustness and an expert evacuation planning team capable to identify possible pitfalls in the percentage distribution.

Appendix A

On parallel computing

A.1 Current scheme

In order to handle the high computational cost emanating from complex agents, a scalable parallel computing extension based on balanced task decomposition is implemented [6].

For the parallelization a distributed memory model using the MPI library is used. The domain is partitioned and distributed with respect to the number of agents. The partitioning is performed using a 2D tree in order to provide each CPU with approximately an equal number of agents. In order to keep continuity in the domain and consider interactions happening with agents in other processes overlapping regions are created. As the simulation progresses agents migrate from one process to another and after several time-steps once load imbalance occurs re-partitioning is performed

Preliminary scalability analysis, using 2 million agents in a $18km^2$ urban environment, is performed attaining 94.8% strong scalability up to 2048 cores, see table A.1. Strong scalability defined as $\frac{\left(\frac{T_m}{T_n}\right)}{\left(\frac{n}{m}\right)}$ with T_k being the run time with k number of CPU cores and $n \geq 2m$. Making use of the presence of disconnected sub-graphs in G_{dep} to implement embarrassing parallelization is not possible due to several reasons, e.g. the number of disconnected sub-graphs is usually much lesser than the required number of CPUs, the links in G_{dep} can change rapidly.

Although the disconnected segments of the graph, if any, allow the introduction of embarrassingly parallel models, they are not robust or computationally inexpensive to be considered for the HPC extension.

A.2 Hybridization

Preliminary tests on the hybridization of the code are performed. This hybridization is accomplished through mimicking the thread local storage strategy, by providing controlled

number of cores	run-time (s)	strong scalability (%)
512	1873.456	
1024	916.2538	102.2
2048	483.2215	94.8

Table A.1: Parallel scalability

access to shared memory regions, see Fig. A.1, and mutually exclusive execution of code through the use of #pragma omp critical in thread unsafe regions of the code.

```
class _Agent{
   private:
        _AgentObject agent_member;
   ...
}

```

(a) Agent with thread unsafe object member

```
//Multi Thread unsafe object

class _AgentObject {
   private:
        static SubObjectType shared_member; //Memory consuming object made
                                            //static to be shared by all agents
                                            //in the MPI PROCESS
   ...

   public:
        void UseSharedMember(){
                shared_member.UseUnsafeFunction();//Multi-thread unsafe function
                                                  //with race conditions
                return;
        }
   ...
}
```

(b) Thread unsafe implementation

```
//Multi Thread safe object

class _AgentObject {
   private:
        static std::vector<SubObjectType> shared_member; //vector of memory consuming object made
                                                         //static (size=max_num_threads)
                                                         //to be shared by all agents in
                                                         //in current PROCESS THREAD
   ...

   public:
        void UseSharedMember(){
                int thread_num   = omp_get_thread_num();
                shared_member[thread_num].UseUnsafeFunction();//Multi-thread safe function
                                                              //through the use of Thread-local
                                                              //storage
                return;
        }
  ...
}
```

(c) Thread safe implementation through pseudo local storage

Figure A.1: Pseudo local storage example

Appendix B

Miscellaneous algorithms

B.1 Topological information graph condensing algorithm

In order to be able to condense important topological information such as the time history of the road usage, the average speed of pedestrians or cars, the agent count per regions, etc. an algorithm that doesn't produce a large impact in the performance of the code and scalable for large number of evacuees had to be designed and implemented. The responsibility of the data condensing is distributed to every process. Each process is in charge of providing summarized data of their active agents. This process can be executed from two perspectives from the agent point of view where each agent contributes its information to the graph, or from the process point of view where each process loops through the active agents list and extracts their information and condenses it. Each of these modes of data condensation have their advantages and disadvantages, depending from the goal and perspective This summarized data is condensed to the each process's individual copy of the topological graph, see 3.1.2. Then, through the use of MPI messages depending on the use case, these local copies of the graphs are condensed into a single process (MPI Gather equivalent), output collectively (collective write) or condensed and distributed among the processes (MPI Allgather equivalent) for their further use in tasks such as collective path planning, or statistical information collection. Pseudo code of this process can be found in algorithm B.1.

The data structure sharing capabilities through messages is built in as part of the data structure providing a developer friendly environment to accomplish this task.

B.2 No obstacle in between

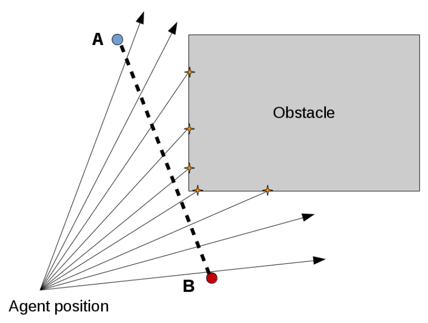
Figure B.1 presents a sketch showing the process of identifying if any obstacle lays in between a line segment and the agent. The agent uses its vision to identify obstacles in

Algorithm B.1 Topological information condensation

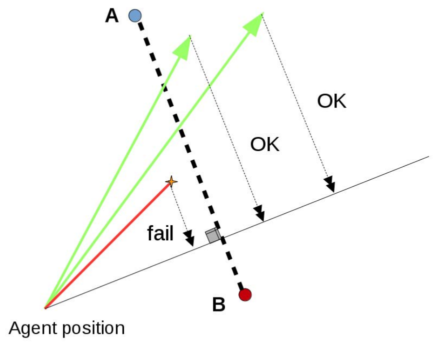
Input: p, target agent position, q quantity to be condensed, $graph$, data structure representing the graph (nodes, and connectivity, edges),$tree$, kd-tree of the topological graph nodes position, l, maximum distance between nodes.

Output: $graph$ with the information of the target agent condensed.

```
//populates $nodes$ with the nodes found through a kd tree search in a square region centered in $p$ with side length $l$
$tree$.get_nodes_in_range($p$, $l$, $nodes$);
$n1 = n2 = pn1 = pn2 = NULL$
$min\_dist = min\_prob\_dist = INF$
$found = false$
for $node$ in $nodes$
    //Populates $neigh$ with the $node$ neighbors
    $graph.neighbors(node, neighbors)$
    $num\_neigh = neighbors.size()$
    if($num\_neigh$ and length($node.position() - p$)<min_prob_dist and is_visible($node.position(), p$))
        $pn1 = node$
        $pn2 = neighbors.first()$
    for $neigh$ in $neighbors$
        //check if the projection of the point on the edge lays between
        //the two nodes and if that point is visible
        //*visibility is only checked in the no earthquake damages case
        if (proy_on_edge($p$,$node$,$neigh$,$p_{proy}$) and is_visible($p$,$p_{proy}$)
        and length($p_{proy} - p$)<min_dist)
            $n1 = node$
            $n2 = neigh$
if $n1 \neq NULL$
        $add\_value(q, n1, n2)$
        **return** $true$
//value added to a probable edge
if $pn1 \neq NULL$
        $add\_value(q, pn1, pn2)$
        **return** $true$
//value could not be added
**return** $false$
```



(a)



(b)

Figure B.1: No obstacle in between sketch

its line of sight, creating from it its visual boundary as presented in section 5.2.1. The extracted visual boundary, or horizon, provides information on the length of each of the rays right before reaching the obstacle. The process to identify if there is any obstacle in between two arbitrary points with the information provided by the horizon is as follows:

- Create a line A-B using the two arbitrary points needed set as the threshold to identify the existence of obstacles in between them (A, B)
- Find a line perpendicular to A-B passing through the origin (agent's position) and measure the distance between the origin and line A-B, *d*
- Project every ray on the perpendicular line, if their distance to the origin is less than the distance *d* and obstacle one of the rays meets and obstacle thus an obstacle existing between A-B and the Origin. If all the rays pass the test there is no obstacle in between

Appendix C

C.A. Tuning parameters, equations and results

The following abraviations are used in the tables:

LB Lower boundary

UB Upper boundary

PS Perturbation size

IV Initial value

VU Value used

PDF Probability density function

C.1 Car-car interaction tuning

Name	LB	UB	PS	IV
τ	1.0	10.0	0.5	5
interaction radius	5.0	40.0	1.0	35
max visibility radius	5.0	60.0	1.0	60
acceleration	0.4	2.0	0.1	0.5
road width	2.5	5.0	0.5	4.5
mean speed 1	8.0	14.0	0.5	12
S.D. speed 1	1.0	10.0	0.5	10
mean speed 2	8.0	14.0	0.5	12
S.D. speed 2	1.0	10.0	0.5	10
speed low limit	5.0	10.0	0.5	5
speed up limit	12.0	18.0	0.5	15
$close\_distance$	2.0	10.0	0.5	5
$far\_distance$	5.0	15.0	0.5	10
$speed\_factor\_close$	0.1	0.4	0.05	0.2
$speed\_factor\_far$	0.1	0.6	0.05	0.5

Table C.1: car-car interaction tuning parameters

Name	VU	Description
τ	3.5	ORCA parameter
interaction radius	36	max. distance to agents to be considered (m)
max visibility radius	60	max. possible sight distance (m)
acceleration	0.6	acceleration(m/s^2)
road width	4.7	maximum usable road width (m)
mean speed 1	11	population 1 speed's PDF mean (m/s)
S.D. speed 1	9	population 1 speed's PDF standard deviation (m/s)
mean speed 2	11.5	population 2 speed's PDF mean (m/s)
S.D. speed 2	9.5	population 2 speed's PDF standard deviation (m/s)
speed low limit	5.5	speed's PDF truncate value, lower threshold (m/s)
speed up limit	14	speed's PDF truncate value, upper threshold (m/s)
$close\_distance$	3.5	see Algo. 4.6
$far\_distance$	10.5	
$speed\_factor\_close$	0.2	
$speed\_factor\_far$	0.5	

Table C.2: car-car interaction parameters value

C.1.1 Error function

$$r(x) = \begin{cases} 22.129077877 \cdot e^{-20.4953636246 \cdot x} & \text{if } x > 0.2 \\ 14.3 & \text{if } x \leq 0.2 \end{cases}$$

C.2 Car-pedestrian interaction tuning

C.2.1 Cars

Name	LB	UB	PS	IV
speed mean	4	20	1	8
S.D. speed	4	10	1	2
speed low limit	2	8	1	4
speed up limit	10	20	1	12
τ	1	10	0.5	2
interaction radius	3	20	1	5
acceleration	0.5	4	0.5	2

Table C.3: car-pedestrian interaction parameters tuning, car

Name	VU	Description
speed mean	9	speed's PDF mean (m/s)
S.D. speed	2	speed's PDF standard deviation (m/s)
speed low limit	4	speed's PDF truncate value, lower threshold (m/s)
speed up limit	13	speed's PDF truncate value, upper threshold (m/s)
τ	1	ORCA scheme parameter (s)
interaction radius	5	max. distance to agents to be considered (m)
acceleration	2.5	acceleration(m/s^2)

Table C.4: car-pedestrian interaction parameters value, car

C.2.2 Pedestrians

Name	LB	UB	PS	IV
speed mean 1	0.6	2	0.2	1
speed mean 2	0.6	2	0.2	1
speed low limit	0.5	1	0.2	0.5
speed up limit	2	2.5	0.2	2.5
$comfort\_ra$	0.25	1	0.2	0.25
τ	1	10	1	2
interaction radius front	2	10	1	4
interaction radius back	2	10	1	3
acceleration	0.4	2	0.1	0.5

Table C.5: car-pedestrian interaction parameters tuning, pedestrians

Name	VU	Description
speed mean 1	1.8	population 1 speed's PDF mean (m/s)
speed mean 2	1.8	population 2 speed's PDF mean (m/s)
speed low limit	0.7	speed's PDF truncate value, lower threshold (m/s)
speed up limit	2.1	speed's PDF truncate value, upper threshold (m/s)
$comfort\_ra$	0.45	see Algo. 4.5 (m)
τ	4	ORCA scheme parameter (s)
interaction radius front	3	max. distance to agents to be considered in the front (m)
interaction radius back	2	max. distance to agents to be considered in the back (m)
acceleration	0.7	acceleration(m/s^2)

Table C.6: car-pedestrian interaction parameters value, pedestrians

Name	VU	Description	
$close\_agent\_min\_distance$	1.2	(m)	see Algo. 4.5
$crowd\_high$	12	# of people	
$crowd\_mid$	7	# of people	
tau_factor	1	-	

Table C.8: car-pedestrian interaction parameters value, pedestrian collision avoidance

C.2.3 Pedestrian collision avoidance

Name	LB	UB	PS	IV
$close\_agent\_min\_distance$	0.5	5	0.5	0.7
$crowd\_high$	10	20	2	10
$crowd\_mid$	5	10	2	5
tau_factor	1	10	1	1

Table C.7: car-pedestrian interaction parameters tuning, pedestrian collision avoidance

C.2.4 Cars collision avoidance

Name	LB	UB	PS	IV
$close\_distance$	3	8	2	5
$far\_distance$	8	15	1	10
$speed\_factor\_far$	0.1	0.5	0.1	0.2
$speed\_factor\_close$	0.2	0.8	0.1	0.5

Table C.9: car-pedestrian interaction parameters tuning, car collision avoidance

Name	VU		Description
$close\_distance$	3	(m)	see Algo. 4.6
$far\_distance$	10	(m)	
$speed\_factor\_far$	0.1	%	
$speed\_factor\_close$	0.5	%	

Table C.10: car-pedestrian interaction parameters value, car collision avoidance

C.2.5 Error function

$$r(x) = \begin{cases} (-0.7932416441 * log(x) + 4.683592813) & \text{if } x > 0.01 \\ 7 & \text{if } x \leq 0.01 \end{cases}$$

Bibliography

[1] Fraser, S.; Leonard, G.S.; Matsuo, I. and Murakami, Tsunami evacuation: Lessons from the Great East Japan earthquake and tsunami of March 11th 2011, GNS Science Report, 2011

[2] Cosenza B. Cordasco G. De Chiara R., Scarano V.: Distributed Load Balancing for Parallel Agent-Based Simulations: In: 19th International Euromicro Conference on Parallel, Distributed and Network-based Processing, pp. 62-69, 2011

[3] L. A. Melgar, L. Wijerathne, M. Hori, T. Ichimura, and S. Tanaka. On the Development of an MAS Based Evacuation Simulation System: Autonomous Navigation & Collision Avoidance, PRIMA 2013: Principles and Practice of Multi-Agent Systems, Springer, Lecture Notes in Computer Science, pp 388-395, 2013

[4] Laubenbacher, Reinhard and Jarrah, AbdulS. and Mortveit, HenningS. and Ravi, S.S., Agent Based Modeling, Mathematical Formalism for, Computational Complexity, Springer New York, 2012, pp 88-14

[5] Muneo Hori, Tsuyoshi Ichimura: "Current state of integrated earthquake simulation for earthquake hazard and disaster", J. of Seismology, pp. 307-321, 2007.

[6] M. L. Wijerathne, L. A. Melgar, M. Hori, T. Ichimura, and S. Tanaka. HPC enhanced large urban area evacuation simulations with vision based autonomously navigating multiagents. Procedia Computer Science, (1515-1524), 2013.

[7] N. Takahashi, Y. Kaneda, Y. Inazawa, T. Baba and M. Kikkojin. Tsunami inundation modelling of the 2011 tohoku earthquake using three-dimensional building data for Sendai, Miyagi prefecture, Japan. Tsunami Events and Lessons Learned, Advances in Natural and Technological Hazard Research, Springer, (35):89-98, 2014.

[8] Xue M. and Ryuzo O. Examination of vulnerability of various residential areas in china for earthquake disaster mititgation. In the 9th International conference on urban earthquake engineering/4th Asia conference on earthquake engineering, pages 1931-1936, Tokyo, 2012

[9] Theodore V. Galambos, Bruce Ellingwood: “Serviceability Limit States: Deflection”, Journal of Structural Engineering, ASCE, pp. 67-84, 1986.

[10] Richard W. Bohannon. Comfortable walking speed of adults aged 20-79 years: reference values and determinants. Age and Ageing, 26:15-19, 1997.

[11] T. F. Nichols and T. R. Powers. Moonlight and night visibility. U.S. Army Training Center Human Research Unit, Presidio of Monterey, California, January 1964. 11) Use-ip ltd. Lux light level chart. Pdf.

[12] Honeywell. Emergency lighting design guide. Pdf, February 2014.

[13] Arizona transportation department, City of Glendale. Street lighting manual. Pdf, 2006 14) M. J. Ouellette, M.S. Rea: “Illuminance Requirements for Emergency Lighting”, Journal of the Illuminating Engineering Society, pp. 37-42, 1989.

[14] Beeson, Jong, and Kuipers, Towards autonomous topological place detection using the Extended Voronoi Graph. IEEE International Conference on Robotics and Automation (ICRA-05), pp. 4373 – 4379, 2005.

[15] Maddegedara Lalith, Leonel A. Melgar, Muneo Hori, Tsuyoshi Ichimura, Seizo Tanaka, On the development of multi agent system for large urban area evacuation with autonomous navigation, Journal of Japan Society of Civil Engineers, Ser. A2 (Applied Mechanics (AM)), Vol. 69, No. 2, I_447-I_456, 2013.

[16] Dhulam R., Lalith M., Hori M., Ichimura T. and Tanka S. A study on effectiveness of using officials for reducing pre-evacuation time in a large area based on multi agent simulations. In 9th International conference on urban earthquake engineering / 4th Asia conference on earthquake engineering. Pages 1658-1665, Tokyo, 2012.

[17] Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha Cedric Pradalier, Roland Siegwart, and Gerhard Hirzinger, Reciprocal n-body collision avoidance , Robotics Research: The 14th International Symposium ISRR, Springer Tracts in Advanced Robotics, vol. 70, Springer-Verlag, May 2011, pp. 3-19.

[18] Takashi Saito and Hiroshi Kagami, *Simulation of evacuation behavior from tsunami untilizing multi agent simulation*, Proceedings ot the 13-th World Conf. on earthquake Engineering, Vancouver,2004.

[19] A. Nash, K. Daniel, S. Koenig and A. Felner,*Theta*: any angle path planning on grids*, Proceedings of the AAAI Conference on Artificial Intelligence, 2007, pp. 1177–1183.

[20] M. Mori and H. Tsukaguchi, A new Method for Evaluation of Level of Service in Pedestrian Facilities, Transp. Res.-A Vol. 21 A, No. 3, pp. 223-234, 1987

[21] U. Weidmann, Transporttechnik der Fussgänger, Transporttechnische Eigenschaften des Fussgängerverkehrs (Literturauswertung), Schriftenreihe des IVT Nr. 90, Zweite, ergänzte Auflage, Zurich, Marz 1993 (109 Seiten)

[23] MULTIAGENT SYSTEMS A Theoretical Framework for Intentions, Know-How, and Communications Munindar P. Singh, Springer-Verlag Lecture Notes in Computer Science, Volume 799 ISBN 0-387-58026-3

[24] Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations Cambridge University Press New York, NY, USA ©2008 ISBN:0521899435

[24] S. Gwynne, E. Galea, M. Owen, J. Lawrence, and L. Filippiddis, *A Review of the Methodologies Used in Evacuation Modeling. Fire and Materials* , 1999, 23, pp. 383–388.

[25] D. Helbing, I. J. Farkas, P. Molnar, T. Vicsek, *Simulation of pedestrian crowds in normal and evacuation situations*, Pedestrian and Evacuation Dynamics, Springer, 2002, pp. 21–58.

[26] M. Hori, H. Miyajima, Y. Inukai and K. Oguni, *Agent simulation for prediction of post-earthquake mass evacuation*, Journal of Japan Society of Civil Engineers, Ser. A, 2008, vol. 64, pp. 1017–1036.

[27] R. Dulam, M. Lalith, M. Hori, T. Ichimura and S. Tanaka, Development of HPC enhanced multi agent simulation code for tsunami evacuation, Journal of Applied Mechanics, JSCE, 2012, vol. 15.

[28] M. L. L. Wijerathne, Muneo Hori, Tsuyoshi Ichimura, Seizo Tanaka, *Parallel Scalability Enhancements of Seismic Response and Evacuation Simulations of Integrated Earthquake Simulator*, High Performance Computing for Computational Science - VECPAR 2012, Lecture Notes in Computer Science Volume 7851, 2013, pp 105-117.

[29] Helbing, D., and P. Molnar. Social force model for pedestrian dynamics. Physical Review E51:4282-7. 1995

[30] V.M. Predetchenskii and A. I. Milinskii, Planning for Foot Traffic Flow in Buildings, National Bureau of Standards, United States Department of Commerce, Amerind Publishing Co. Pvt Ltd, New Delhi 1978

[31] Jun Zhang, Pedestrian fundamental diagrams: Comparative analysis of experiments in different geometries, Forschungszentrum Jülich GmbH, 2012

[32] S. L. Dhingra, Ishtiyaq Gull, Traffic Flow Theory Historic al Research Perspectives, Transportation Research Circular E-C149: 75 Years of the Fundamental Diagram for Traffic Flow Theory, Transportation Research Board 2011

[33] Ghoshray, S.; Yen, K.K., "A comprehensive robot collision avoidance scheme by two-dimensional geometric modeling," Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on , vol.2, no., pp.1087,1092 vol.2, 22-28 Apr 1996

[34] Volkan Isler, Dengfeng Sun and Shankar Sastry, Roadmap Based Pursuit-Evasion and Collision Avoidance, Roadmap Based Pursuit-Evasion and Collision Avoidance, In: Proc. Robotics: Sci. Sys., RSS 2005

[35] Emiliano Cristiani, Benedetto Piccoli, Andrea Tosin, Multiscale Modeling of Pedestrian Dynamics, Chapter4, Springer 2014

[36] Stephen Jacob, “Multi agent simulation of tsunami triggered evacuation for earthquake damaged environments”, MSc degree thesis, The University of Tokyo, 2014

[37] S. L. Dhingra, “Traffic Flow Theory Historical Research Perspectives” Transportation Research Circular E-C149: 75 Years of the Fundamental Diagram for Traffic Flow Theory, Transportation Research Board of the National Academies, 2011

[38] TANI Kenji, Maps and GIS Data of Contour,Altitude and Population related to the Great East Japan Earthquake, http://ktgis.net/tohoku_data/index_e.html, visited: February 2015

[39] Tsunami Damage Mapping Team, Association of Japanese Geographers, Maps of the Area hit by the Tsunami of 11 March 2011, Northeast Japan, http://danso.env.nagoya-u.ac.jp/20110311/map/574152Ishinomaki3.jpg, visited: February 2015

[40] Rithika Dulam, Enhancement of Multi Agent Simulation with Smart Agent Interaction and High Performance Computing, Master Thesis, The University of Tokyo 2012

[41] Stephen Jacob, Leonel Aguilar, Lalith Wijerathne, Muneo Hori, Tsuyoshi Ichimura, Seizo Tanaka, Agent based modeling and simulation of tsunami triggered mass evacuation considering changes of environment due to earthquake and inundation, Journal of Japan Society of Civil Engineers, Applied Mechanics Special Issue, 2014, Journal of Japan Society of Civil Engineers, Ser. A2 (Applied Mechanics (AM)), Vol. 17, No. 2, pp 671-680, 2014