

FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

ESTÊVÃO SMANIA TESTA

MODELO DE ESTIMAÇÃO DE MULTIDÕES PRA CENÁRIOS DE EMERGÊNCIA

Porto Alegre

2018

PÓS-GRADUAÇÃO - *STRICTO SENSU*



Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
GRADUATE PROGRAM IN COMPUTER SCIENCE

CROWD ESTIMATION MODEL FOR EMERGENCY SCENARIOS

ESTÊVÃO SMANIA TESTA

Dissertation presented as partial requirement
for obtaining the degree of Master in
Computer Science at Pontifical Catholic
University of Rio Grande do Sul.

Advisor: Prof. Soraia Raupp Musse

Porto Alegre
2018

MODELO PARA ESTIMATIVA DE PARÂMETROS DE MULTIDÕES EM SITUAÇÕES DE EMERGÊNCIA

RESUMO

Planos de evacuação têm sido historicamente usados como uma medida de segurança para a construção de edifícios. Os simuladores existentes requerem ambientes 3D totalmente modelados e tempo suficiente para preparar e simular cenários. Uma vez que a quantidade de pessoas pode mudar ao longo do tempo, várias simulações são frequentemente necessárias para gerar um plano de evacuação otimizado. Neste documento é apresentada uma nova abordagem para estimar os dados resultantes de um dado cenário de evacuação sem simula-lo de fato. Para tal o ambiente é dividido o ambiente em salas modulares com configurações diferentes, em um estilo divisão e conquista. Em seguida, uma rede neural artificial é treinada para estimar os dados desejados de uma sala sozinha. Após coletar os dados estimados de cada sala, uma heurística capaz de agregar informações por sala é desenvolvida para que o ambiente completo possa ser devidamente estimado. Esse método apresenta erros dentro da margem de 30% quando comparado o tempo de evacuação em um ambiente real e complexo. Além disso, não é necessário modelar o ambiente 3D, aprender como configurar um simulador de multidões e o tempo computacional para estimar é instantâneo quando comparado ao melhor caso de um simulador de multidões.

Palavras Chave: Evacuação de Multidões, Simulação de Humanos Virtuais, Aprendizado de Máquina, Segurança, Multidões.

CROWD ESTIMATION MODEL FOR EMERGENCY SCENARIOS

ABSTRACT

Evacuation plans have been historically used as a safety measure for the construction of buildings. The existing simulators require fully-modeled 3D environments and enough time to prepare and simulate scenarios. Since the amount of people in a given simulated scenario can change over time, several simulations are often required in order to generate an optimal evacuation plan. With that in mind, we present in this paper a novel approach to estimate the resulting data of a given evacuation scenario without actually simulating it. For such, we divide the environment into modular rooms with different configurations, in a divide-and-conquer fashion. Next, we train an artificial neural network to estimate all required data regarding the evacuation of a single room. After collecting the estimated data from each room, we developed a heuristic capable of aggregating per-room information so the full environment can be properly evaluated. Our method presents errors within the 30% margin when compared to evacuation time in a real and complex environment. In addition, it is not necessary to model the 3D environment, learn how to use and configure a crowd simulator, and the computational time to estimate is instantaneous when compared to a best case real-time crowd simulator.

Keywords: Crowd Egress, Virtual Human Simulation, Machine Learning, Safety, Crowds.

LIST OF FIGURES

3.1	Comparison between the <i>ORCA</i> 's showcase (top) and our simulation (bottom). Agents are positioned in a circular pattern and moves to the opposite end of the circle while avoiding collisions with others.	27
3.2	Comparison between the <i>ORCA</i> 's showcase (top) and our simulation (bottom). 41 agents are positioned forming the word "RVO" and moves down the screen to form the word "UNC" while avoiding collisions with others.	28
3.3	Footage from a group of agents making a turn in the <i>Rounding Corners</i> test.	29
3.4	<i>Counter flow</i> scenario configuration, following IMO's specification.	29
3.5	<i>Exit flow</i> scenario configuration, following IMO's specification.	30
3.6	<i>Exit route allocation</i> scenario configuration, following IMO's specification.	31
3.7	Error frequency of the validation cases with neural networks with one hidden layer with 6 neurons. The number of occurrences of error sizes are grouped in bars of 10% error size, with a percentage indicating how much of the validation cases are in it, also the average error size of all cases and its correspondent standard deviation are shown.	34
3.8	Search for a better network model. Attempts with 6, 50, 200, 400 and 500. Plotted number of cases bellow 10% error in the individual validation.	35
3.9	Error frequency of the validation cases with the improved neural networks. The number of occurrences of error sizes are grouped in bars of 10% error size, with a percentage indicating how much of the validation cases are in it, also the average error size of all cases and its correspondent standard deviation are shown.	36
3.10	Environment editor main window, control panel and canvas regions and a room are highlighted in red, black arrows indicate each rooms connections to the others.	40
3.11	Environment editor's room values window, it shows the values of the parameters of the selected room for the user to modify it to design the environment.	41
4.1	A environment with the scenario Sim 8 modeled in the environment editor.	44
4.2	Scenarios modeled in the Simulator developed in Unity.	45
4.3	Obtained results when testing 20 environments. The percentage of errors are shown for total evacuation time tt_e and for average time \bar{t}_e metrics for each environment.	47
4.4	Nine identical rooms placed in sequence in order to be estimated.	48
4.5	Absolute relative evacuation time (tt) error sizes for each of the propagation cases, ordered by the number of rooms (14 cases, starting with 3 rooms and ending with 29 rooms).	49
4.6	Architecture of the nightclub used as base for simulation and estimation.	50
4.7	Modeling of SM night club in the Environment Editor.	51

4.8 Errors in percentage as resulted from the comparison between the estimation and simulation of a practical example. 52

CONTENTS

1	INTRODUCTION	15
2	RELATED WORK	17
2.1	FUNDAMENTALS OF CROWD SIMULATION	17
2.1.1	COLLECTIVE BEHAVIORS	17
2.1.2	NAVIGATION MESH (<i>NAVMESH</i>)	18
2.1.3	OPTIMAL RECIPROCAL COLLISION AVOIDANCE (<i>ORCA</i>)	19
2.1.4	NEURAL NETWORKS IN CROWDS	20
2.2	SIMILAR WORKS	21
2.2.1	CROWD SIMULATION	21
2.2.2	EVACUATION PLANNING	22
2.2.3	CROWD LEARNING	22
3	THE MODEL	25
3.1	CROWD SIMULATOR	25
3.1.1	MODEL VALIDATION	26
3.1.2	COMPARING WITH <i>ORCA</i> SHOWCASES	27
3.1.3	VALIDATION ACCORDING TO IMO	28
3.2	ESTIMATION OF ROOMS PARAMETERS BASED ON ANNS	31
3.2.1	DATABASE CREATION	32
3.2.2	TRAINING AND VALIDATION	34
3.3	ENVIRONMENT ESTIMATION BASED ON ANNS AND HEURISTICS	35
3.3.1	ENVIRONMENT EDITOR	39
4	EXPERIMENTAL RESULTS	43
4.1	EVALUATING ENVIRONMENTS WITH VARIED POPULATIONS	43
4.2	INVESTIGATING REPLICATED ROOMS	47
4.3	TESTING A PRACTICAL EXAMPLE	49
5	FINAL REMARKS	53
	REFERENCES	55

1. INTRODUCTION

As new buildings are designed and constructed by engineers and architects, evacuation procedures to assure the needed safety standards are a major concern. Evacuation drills are usually used to analyze and evaluate pre-defined evacuation plans, but despite having strong similarities with real emergency scenarios [4], they still present significant ethical, practical, and financial challenges to researchers [21].

Crowd simulation has been a more viable approach for research purposes. It can be defined as the process of simulating the movement of large amounts of entities, or crowds, in a known environment. The different ways in which a crowd can behave has been the research interest of many researchers for almost thirty years. These researchers come from a wide range of fields, including architecture, computer graphics, physics, robotics, safety engineering, training systems, psychology and sociology [42] [36].

When simulating crowds, a set of parameters that reproduce coherently the desired scenario should be considered. Cassol et al. [7] enumerated them as aiming to represent: *i) Environment's physical structure*: dimensions, number of floors, number of rooms, location of exits and stairs; *ii) Environment functionality*: whether people act as though they were in an office, hospital, school, airport, stadium or arena; *iii) Population data*: number and spatial distribution of people in the environment, age, gender, relationships among them, knowledge about the environment; and *iv) Environment condition in events*: factors that may affect the navigability in the scenario, such as time (day or night), smoke, fire, or heat.

As of now, in the evacuation planning scope, crowd simulation can be used to search for better evacuation plans given a parametrized environment. Cassol et al. [7] used *CrowdSim*, a crowd simulator validated and tested with real scenarios [6] [8] [17] to create and gather data from simulations, then used CMA-ES, an evolutionary strategy [24] [23], to vary the population data so that different portions of the crowd follow different routes. The data was then evaluated using the *ep* metric [6] in order to find which configuration of routes was better for the evacuation. Indeed, this metric aims to depict a way to evaluate a given evacuation plan (more details in [7]). Although the idea is simple, when considering all the possible variations a certain environment can have, there will be a large quantity of simulations to be executed, growing exponentially as more details are added in the environment.

This document presents a method whose goal is to estimate the data for a certain evacuation scenario without simulating it. In order to accomplish with this goal, we divided the environment to be estimated into a set of connected rooms. Our main hypothesis is that by estimating evacuation information about each room, we will be able to estimate data for the entire environment. The data of each room is estimated using Artificial Neural Networks (ANN) trained with data collected from many simulations using a crowd simulator, also developed in this document. Furthermore, we defined an heuristic to provide the data estimation for more complex environments composed by a set of rooms. Later, we analyzed the behavior of the estimator when presented with worst case

scenarios which we suspected could be problematic. We finished with estimating a real environment and evaluating the result obtained with a simulated result.

The estimation of crowd evacuation scenarios using ANN is the main contribution of this work, since it is a new approach for Crowd Behavior as far as we know. We did not find any other work that used machine learning in crowd simulation area with similar purposes.

This paper is organized as follows: in Section 2 we present works related to the developed project. Then, in Section 3 we describe our proposal towards the goal of this work. In Section 4 we describe some experimental results obtained with our model, while in Section 5 we discuss some final considerations and future works.

2. RELATED WORK

The study and modeling of crowd traffic and behavior, environment and evacuation characteristics is vital to the use of critical spaces [15] [34] [37] [39]. Modern hardware and computational methods afford large-scale simulations of such pedestrian crowds in arbitrary environments [42]. In this chapter we focus on the fundamentals and techniques used in this work to simulate and estimate crowds, described in Section 2.1, and in similar works on crowd analysis and simulation area presented in Section 2.2.

2.1 Fundamentals of Crowd Simulation

In this Section we present some concepts of Collective Behaviors, studied to understand behavior phenomenons that occur when people gather in crowds. In addition, we describe some tools and techniques used for simulating agents to behave as real persons in virtual environment, as *NavMesh* and *ORCA*. An introduction to the basic of neural networks focused on crowds is also discussed.

2.1.1 Collective Behaviors

Most crowd behaviors are hard to evaluate and validate due to the difficulty to measure all variables that influence them. For this reason, some crowd behaviors are only evaluated and validated qualitatively. From the observations of behavioral patterns in crowds [14, 25, 26, 27, 28, 41] authors discuss two types of behaviors: inherent and emerging behaviors.

Inherent behaviors are the ones adopted in order to make a person move in the most efficient way possible in an environment. Some persons may present irregularities in these behaviors, like children, because they are still learning about the environment and how to move effectively in it [25]. The main inherent behaviors are:

- *Goal seeking*: people move in an environment in order to arrive at a goal it is seeking;
- *Collision avoidance*: people move avoiding physical contact with obstacles and with others; and
- *Least effort strategy*: people choose trajectories that spend less energy, avoiding turns and selecting the shortest paths. In some cases this conflicts with *collision avoidance*, since in most cases it is needed to change its orientation to avoid to collide with others.

Emerging behaviors are collective ones resulting of the self-organization present in crowds. These behaviors appear because of the non-linear interaction between persons applying inherent behavior and sometimes may lead to collisions and obstructions. The main emerging behaviors are:

- *Lane formation*: When in high density, the movement of groups in contrary directions causes lanes of peoples walking in the same direction. This happens because a person follows the one with the closest velocity in front of him/herself in order to reduce the needed effort to reach his/her goal;
- *Organization prior*: When groups move in contrary directions it is possible to organize the movement beforehand by compacting small groups temporarily to free space for people to cross reducing the effort made; and
- *Speed reduction effect*: As much the people density increases, less is the people speed. This is because of the limited space to move and also to avoid physical contact. The crossing of different flows aggravates this effect.

The following behaviors happen due to the *speed reduction effect*:

- *Arc formation*: When a great number of people moves toward a small passage they form a geometric arc on the entrance. They agglomerate due to the reduced speed and their wish to stay close to the passage;
- *Bottleneck effect*: This effect can be observed on a corridor that at some point narrows. The region before the narrowing will increase in density and people in this area reduce their speed while regions after the narrowing presents decreased density with increased speed;
- *Corner effect*: In high density, people in regions with corners have a reduced speed because the space is not used effectively in these regions; and
- *Shockwaves effect*: In high densities, with reduced speed, people can push others, which is propagated into the crowd, resembling a wave effect.

For the development of our crowd simulator, we implemented the agents to obey the inherent behaviors as they are the main collective behaviors. Emerging behavior rules do not need to be implemented directly, as they should appear if the agents are following the inherent ones.

2.1.2 Navigation Mesh (*NavMesh*)

NavMesh is an abstract data structure used for finding paths in a virtual environment and is the current state-of-the-art in path finding. It appeared for use in the robotics area in 1987 with the name *Meadow mapping* [2], however became a frequently used tool for game development after the year 2000 [44] [40].

As described by Snook [40], *NavMesh* is a polygonal mesh composed of only convex polygons, where adjacent polygons are connected with each other in a graph. It can be viewed as a polygonal carpet covering the polygons of the ambient model, defining the area that is traversable for

agents to know. Paths between the polygons of the *NavMesh* can be found by adapting normal graph search algorithm like A^* , and paths on a single polygon can be found by trivially tracing a straight line between two points on its surface, since the polygons, it is composed of, are convex. Agents following these paths do not need to use expensive collision avoidance algorithms with obstacles since the paths are traced over guaranteed traversable area.

Different versions of *NavMesh* were developed along the time, one example is the *Explicit Corridor Map* [19, 20, 48, 49] a version of *NavMesh* which uses a connected graph of different sized circles instead of polygons as walkable areas. These circles are connected if a straight line can be traced from their centers, and the size of the circles indicates how far away from the center agents can walk in the environment.

In this work we used the Unity implementation of *NavMesh* in the crowd simulator developed to generate the training database. This implementation goes along tools for automatically build the mesh according to the scenario and code to find optimal paths from two points inside this mesh.

2.1.3 Optimal Reciprocal Collision Avoidance (*ORCA*)

ORCA is a protocol for collision avoidance between multiple agents proposed by Van Den Berg et al [46]. The objective of the method is to find a velocity for each agent so that it manages to move through the environment without colliding in an optimal manner, i.e. each agent should consider each other agent's velocity and regulate its movement in order to reduce the least possible its speed, while trying to reach its goal.

The method was developed for being used in robotics collision avoidance. One of its greatest features is that it runs at $\mathcal{O}(n)$ without needing to maintain communication between the robots, just needing them to perceive each others position and velocities. *ORCA* provides that each robot motion is free of collisions for at least a fixed amount of time assuming each robot uses it as collision avoidance protocol.

Agents are represented as a simple circle shape and are holonomic, i.e. can move in any direction without needing to turn around. A possible velocity for an agent A is computed by finding the permitted velocities V_A between agent A and each other agent. To do so it is used the velocity obstacle (VO) between A and each other agent B for the time T :

$$VO_{A|B}^T = \{v | \exists t \in [0, T] :: t.v \in D(p_B - p_A, r_A + r_B)\}, \quad (2.1)$$

where:

v = relative velocity of A with respect to B ,

r_x = radius of agent x ,

p_x = position which agent x is centered, and

$D(p, r)$ = open disc of radius r , centered at position p .

In other words, VO contains all relative velocities of A with respect to B that will result in a collision between them during time T .

Then for any set of velocities V_B , if $v_B \in V_B$ and $v_A \notin VO_{A|B}^T \oplus V_B$, it means that A and B are guaranteed to be collision free for at least T time, which is the set of collision avoiding velocities:

$$CA_{A|B}^T(V_B) = \{v | v \notin VO_{A|B}^T \oplus V_B\}, \quad (2.2)$$

where the operator \oplus denotes the *Minkowski* sum of two sets, i.e. it returns a new set with the sum of the elements of the two sets on the same index.

If $V_A \subseteq CA_{A|B}^T(V_B)$ and $V_B \subseteq CA_{B|A}^T(V_A)$ is true then it means these two are sets of reciprocally collision avoidance velocities. The optimal reciprocal collision avoidance velocity are the pair of such velocities that increases the number of velocities closer to the optimization velocity for each agent:

$$ORCA_{A|B}^T(V_B) = \{v | (v - (v_A^{opt} + u/2))n \geq 0\}, \quad (2.3)$$

where:

v_x^{opt} = the optimization velocity of agent x ,

u = the vector from $v_A^{opt} - v_B^{opt}$ to the closest point on the boundary of the velocity obstacle, and

n = the outward normal of the boundary of the $VO_{A|B}^T$, at point $(v_A^{opt} - v_B^{opt}) + u$.

As *ORCA* is reciprocal, only half of u is applied so that each agent takes half the responsibility to avoid the collision. By carefully choosing the optimization velocity, there will always be a solution within *ORCA* that guarantees the collision free for at least T time.

2.1.4 Neural Networks in Crowds

The use of machine learning and deep learning have been expanded in diverse areas, including crowd analysis and simulations. Artificial neural networks are frequently being used for finding patterns in data and identifying objects, humans and animals in images. In the crowd area, neural networks are frequently being used for counting and estimate the quantity and density of people in an image [3, 9], a difficult problem due to the camera angle and occlusion of persons in a dense crowd.

The basic neural networks known today is the multilayer perceptron [38]. As described in [22], in this architecture, artificial neurons are organized in layers, each layer represents a matrix of weights which are the connections between its neurons and the neurons in the layer before. Weights are used for a matrix multiplication in each layer, with the output of one layer is the input of the following layer, propagated the data along the network. The layers between the input and the output layers are named hidden layers. The learning process in the multilayer perceptron is supervised i.e. it is done by comparing the error from the network result and a "truth" value when having the same input, then the responsibility of each weight over this error is calculated by undoing the matrix multiplications operations over the error value, a process named backpropagating [35], and the network learns by adjusting the value of each weight according its responsibility. In addition, a learning rate is used to prevent the network from learning too much from a single example. Other neural network models uses others diverse operations instead of a simple matrix multiplication, but the general base of the neural network remains the same, i.e. to propagate the input forward the output layer, to measure the output error then backpropagate it to adjust the weights of the network.

About the use of neural networks in the context of crowd simulation topic, there is a great focus in estimating the number of people in a determined environment, as mentioned before. To our best knowledge, there was no attempt at using machine learning techniques to learn or estimate evacuation time, exit flow or any data from emergency events. In this work we explore supervised learning in neural networks to propose a solution to such problem.

2.2 Similar works

Despite the fact that we did not find any method specifically with our goal in the literature, we present some few existing works in the areas related to our project: crowd simulation, evacuation planning and crowd learning. We are not exhaustive in this presentation since none of them are really focused on the problem we want to solve in this dissertation.

2.2.1 Crowd simulation

Several studies were proposed to elaborate ways of simulating crowds in egress scenarios. Besides the already mentioned *CrowdSim* [6] [8] [17] there are other approaches as following briefly described:

- *SAFEgress* (Social Agent For Egress) [11] is a force based approach that models evacuating pedestrians which are able to make their actions according to their knowledge of the environment and their interactions with the social groups and neighboring crowds.

- MIMOSA: Mine Interior Model Of Smoke and Action [29] is a specific application of agent modeling, with the context of a virtual underground coal mine, a fire and smoke propagation model, and a human physiology and behavioral model.
- *BioCrowds* [12] is a biologically-motivated approach which simulates crowds using a space colonization algorithm, recreating several real crowds behaviors such as collision avoidance, speed reduction effect and lane formation.
- Tsai et al. [45] developed a general framework for evacuation simulation using particle swarm optimization (PSO), originally proposed in [32].

Also, the works proposed in [31] [1] [16] make use of cellular automata to reproduce pedestrian behavior and exit selection using a least effort cellular automaton algorithm, in which the motions and goals are probabilistic.

2.2.2 Evacuation planning

As for evacuation planning, the already mentioned Cassol et al. [7] makes use of crowd simulation (*CrowdSim*) and evolutionary strategies (CMA-ES) to search for the best configuration of routes for evacuation in the environment.

Similarly, Garrett et al. [18] used Evolutionary Computation search methods to evolve the placements of exits and other equipments in an effort to minimize the simulated evacuation time of the environment's occupants. The simulation is made using an artificial potential fields model in which exits attract agents and obstacles and other agents repel them.

2.2.3 Crowd learning

Currently, most of the works in the area is on the crowd counting problem [3,9], also called density estimation. For example, Chan et al. [10] created a privacy-preserving system for estimating the size of inhomogeneous crowds in a video, by segmenting it using a dynamic texture motion model, the correspondence of each segmented region's features and the number of people in it is estimated with a Gaussian Process regression. Similarly, Fradi et al. [13] extracted features from videos, and used Gaussian symmetric kernel function to generate a crowd density map, allowing more specific location of potentially crowded area.

Liu et al. [33] is the closest approach to our work, that we found in literature. They make use of artificial neural networks to learn the behavior of simulated crowds with the objective of replacing simulation with estimations, which is basically our same goal. They simulate the crowd in a fixed environment containing mobile parts whose disposition varies for each simulation. The position and rotation of these mobiles parts are used as input to the neural network to obtain the

statistics of speed, number of collisions, and traveling time of each agent for this environment and crowd configuration. While they also use crowd simulation associated to neural networks, the authors propose a local estimation in same environment while we are interested about proposing a global solution where many environment configurations can be estimated. Our scope is currently limited to have rectangular rooms in any complexity environments, as we can see in this work, we tested with huge environments (having 29 rooms) and also with a night club that exists in real life.

3. THE MODEL

In this Section we describe details about our proposed method. First, we detail the development of the crowd simulator used to generate data for the learning process; secondly, we present the process used to validate this simulator; and finally we describe our estimation methodology.

3.1 Crowd Simulator

To develop the crowd simulator we make use of *Unity3D* game engine as a platform and implemented agents endowed with the main inherent collective behaviors as referred in the literature: *i) goal seeking*, *ii) collision avoidance* and *iii) Least effort strategy* behaviors, [14] [26] [27] [28] [25] [41].

A *goal seeking* agent means that the agent moves inside the environment in order to arrive at a goal it is seeking. This behavior was simple to implement, just by keeping a goal position for each agent and moving the agent towards it at each simulation step, was enough to imitate this behavior.

When applying a *Least effort strategy*, the agent moves in trajectories that require less effort, avoiding turns and selecting the shortest paths. This behavior was implemented using path finding over a *NavMesh*, a network of connected 3D planes which represents the navigable area. The *NavMesh* used was the one native of *Unity3D* platform.

When applying a *Collision avoidance* behavior, the agent moves avoiding physical contact with obstacles and with other agents. In part, this behavior is implemented using the *NavMesh* to avoid collision with obstacles. In order to avoid collision with other agents we make use of *ORCA* [46].

In our simulator, we can have big environments having many rooms and agents. Our only restriction is that each room should be a rectangle having only one exit. Indeed, we argue that such restrictions can still work for any environment if we abstract the concept of room for a "rectangular region with one only exit". It means that if we have a different shape in a real environment to be simulated, the region considered in our simulation is one approximate rectangle for each exit. So, for instance a room modeled as an octagonal shape with two exits should be represented through two rectangular rooms, each one having one exit. Clearly, it can generate some errors in comparison to a full simulation of the environment, e.g. spaces that are not used or people that should be located in one region, but in fact they wanted to go to another exit, but we still argue that such simplicity has benefits in terms of modeling and scenario configurations of crowd simulations.

The room parameters defined in our method are:

- *Room width* (meters),
- *Room length* (meters),

- *Exit size* (meters),
- *Input flow* (Agents per second that entry in the room) - it can be zero if no agents entry in this room,
- *Flow Duration* (Duration in seconds defining the period where agents entry in this room) - it can also be zero, and
- *Initial population* (Number of agents which are inside the room at the beginning of the simulation).

As output, we collected the following data for each simulated room i and for each environment e having N rooms:

- *Evacuation total time* tt_i and $tt_e = \sum \frac{tt_i}{N}$ (seconds),
- *Average exit time* \bar{t}_i and $\bar{t}_e = \sum \frac{\bar{t}_i}{N}$ (seconds),
- *Average speed* \bar{s}_i (meters per second) and $\bar{s}_e = \sum \frac{\bar{s}_i}{N}$; and
- *Average density* \bar{d}_i and $\bar{d}_e = \sum \frac{\bar{d}_i}{N}$ (agents per meter).

We designed simulations to be generated using the cited six inputs in evacuation scenarios. We collected data during the generated simulations to create a database of evacuation simulations used for training neural networks later on. This is going to be more detailed in Section 3.2. However, before to describe the usage of simulation to estimate the crowd parameters using ANNs, we present the validation of our simulator in next Section.

3.1.1 Model validation

The validation of our crowd simulation was separated in two steps: firstly, to evaluate our *ORCA* implementation when compared to the original one, and secondly, to validate its similarity with real crowds.

In order to compare with the original *ORCA*, we recreated two of the *ORCA*'s showcases available at the developer's website [47]. Footage of these showcases can be seen in Figures 3.1 and 3.2.

For the real crowd validation we used the same cases used to validate *CrowdSim* [5] which was based on the *guidelines for evacuation analysis for new and existing passenger ships* proposed by the International Maritime Organization (IMO) [30]. This validation is composed of component and qualitative tests. Further details are presented in Section 3.1.3.

3.1.2 Comparing with *ORCA* showcases

Two showcases were created to show the agents behavior using *ORCA* to avoid collision and its capacity to move efficiently while looking natural to the human eye. We recreated the showcases only using visual information from real *ORCA* cases, without knowing the agents parameters (e.g. agents positions or speeds), so we expected similar results but maybe not the same. Nevertheless, a good visual comparison of the behaviors could still be done.

The first showcase consists of an environment with no obstacles and ten agents positioned in a circular pattern, in which each agent is supposed to move towards the opposing end of the circle, while avoiding collision with the others. As the agents move towards the center, the space between them becomes smaller and some of them need to assume non-optimal routes and reduce velocities to avoid the collisions. In both implementations (*ORCA* showcase and ours), agents move successfully towards their goals, reduce their speed and change their courses to avoid collision with others. Both the first showcase and our simulation can be visualized in Figure 3.1.

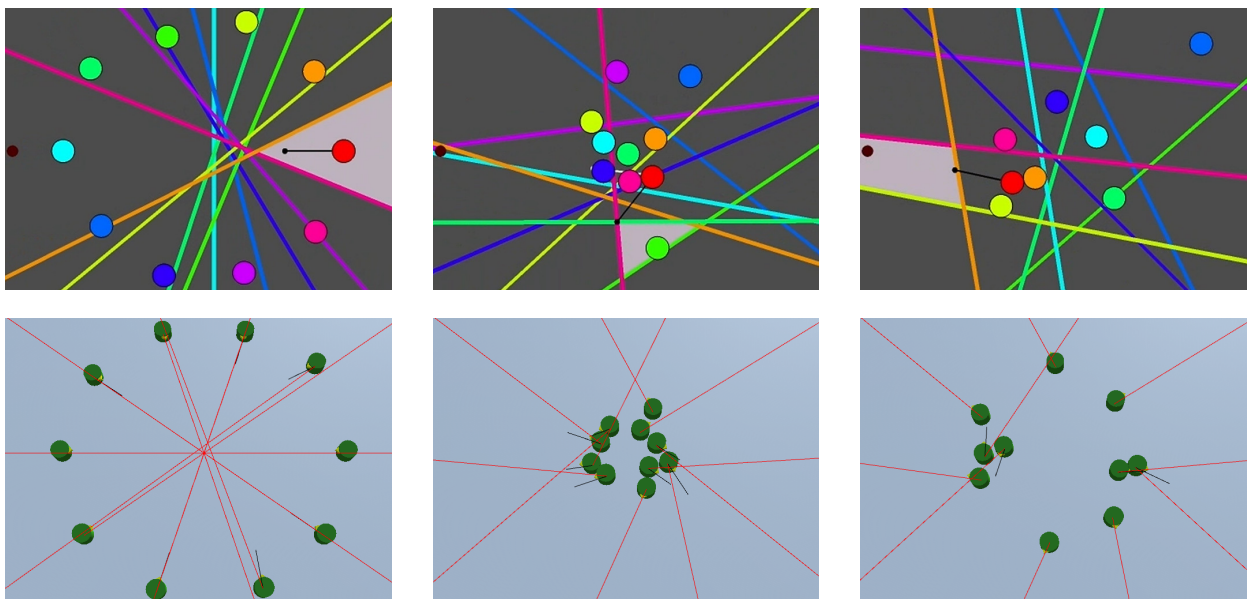


Figure 3.1 – Comparison between the *ORCA*'s showcase (top) and our simulation (bottom). Agents are positioned in a circular pattern and moves to the opposite end of the circle while avoiding collisions with others.

The second showcase consists of 41 agents in three groups, the agents of each group are positioned to form each the letters of the word "RVO". Then they proceed to move down the screen to form the word "UNC". The group forming the letter "R" moves to form the letter "N" at the end of the showcase, the one forming the letter "V" forms the letter "C" and the one from the letter "O" forms the letter "U". Easier movements could be made to complete this transformation, however these specific movements were chosen in order to promote more potential collisions, thus requiring the use of collision avoidance mechanisms, ideal for testing *ORCA*. A comparison showing the showcase and our simulation as well as the movement pattern generated by both simulators can

be visualized in Figure 3.2. In both cases the agents succeeded in moving and forming the other word. /estPlease, notice, that we did not know in details the scenario and parameters used for the showcases, instead we just imitated them based on visual comparison of the results obtained.

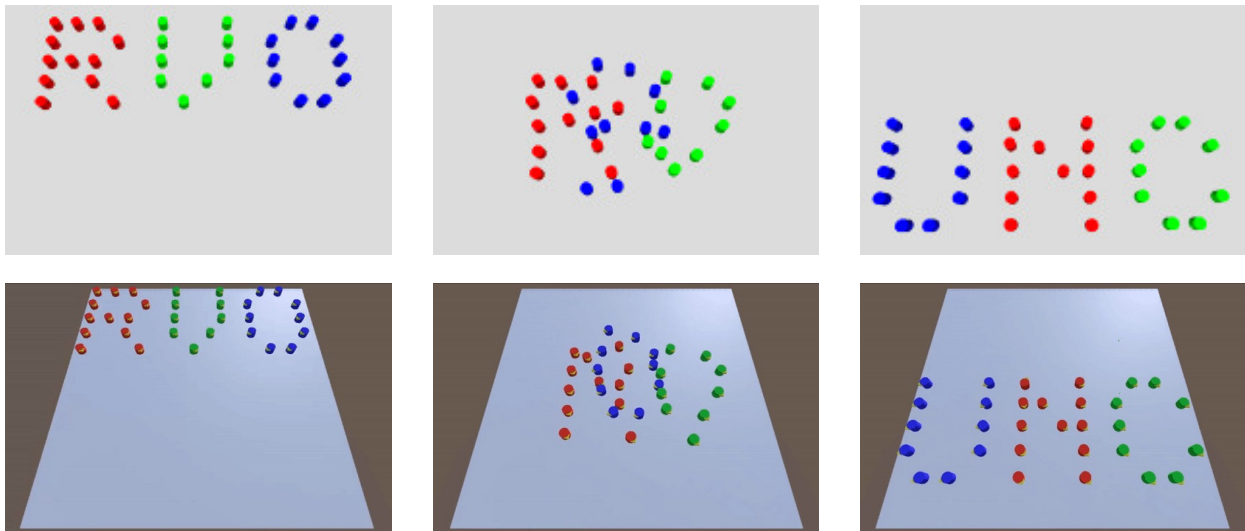


Figure 3.2 – Comparison between the *ORCA's* showcase (top) and our simulation (bottom). 41 agents are positioned forming the word "RVO" and moves down the screen to form the word "UNC" while avoiding collisions with others.

3.1.3 Validation according to IMO

As mentioned before, we propose to validate our simulator using the *guidelines for evacuation analysis for new and existing passenger ships* developed by the International Maritime Organization (IMO) [30]. This validation is composed of component and qualitative tests. Component tests are used to check if the tested software performs as intended. Two component tests were conducted: *Maintaining set walking speed* and *Rounding Corners*. Qualitative tests are concerned with the nature of predicted human behavior with informed expectations from observed situations, as they demonstrate the capability of the model to produce realistic behaviors. Three qualitative tests were realized: *Counter flow*, *Exit flow* and *Exit route allocation* tests. They are detailed in next Sections.

A) Maintaining set walking speed: This test consists of simulating a single agent moving in a straight line free of obstacles at 1m/s and it is successful if the agent walks 10 meters in 10 seconds. In our simulation the agent succeeded by arriving at the end of the 10 meters corridor at time 9.66 seconds.

B) Rounding Corners: This test evaluates the agent's ability to navigate around a corner without penetrating the walls and without overlapping each other. For this we simulated 50 agents moving towards a goal in a corridor with a 90 degrees turn. The corridor is 2 meters wide with a 10

meters long segment before the turn and another segment after the turn, the area of the turn is a 2x2 meters square. During the test we have detected no agents outside the navigation area, which means no agent penetrated obstacles, and no overlapping between the agents during the turn, so the test was considered a success. An image of the test can be seen in Figure 3.3.

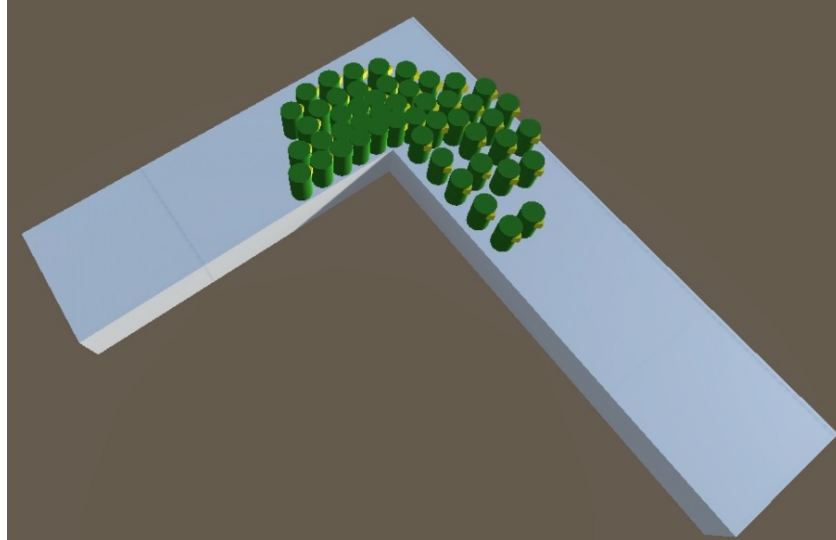


Figure 3.3 – Footage from a group of agents making a turn in the *Rounding Corners* test.

C) *Counter flow*: This test consists of moving 100 agents between two rooms of 10x10 meters connected by a corridor of 2 meters wide and of 10 meters length. The test is repeated with a counter flow of 10, 50 and 100 agents. The time it takes for the last agent to enter the second room is recorded and it is expected that the time increases as the people in the counter flow increases. IMO's guidelines for this test can be seen in Figure 3.4. As can be seen in Table 3.1, the recorded time for our simulations increased along with the agents in counter flux, making the test successful.

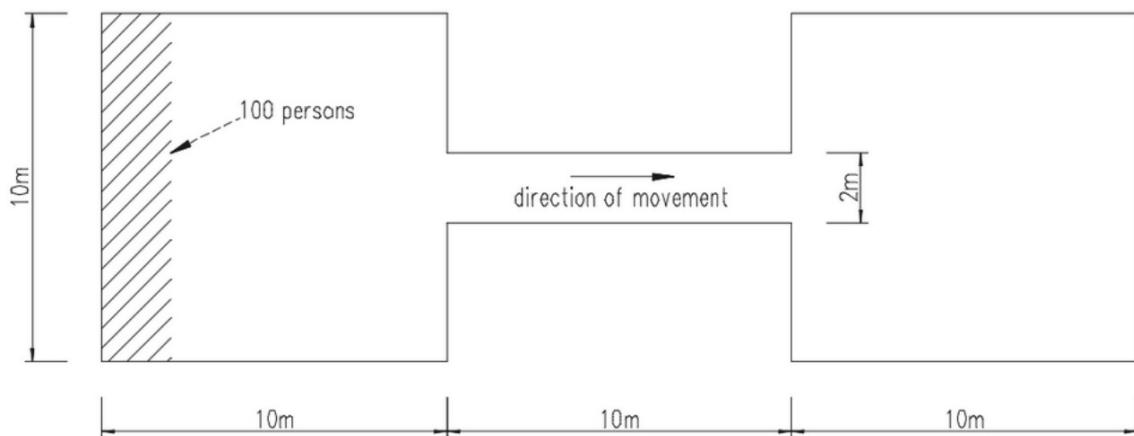


Figure 3.4 – *Counter flow* scenario configuration, following IMO's specification.

Case	Time (seconds)
No counter flux	21 seconds
10 counter flux	30 seconds
50 counter flux	45 seconds
100 counter flux	55 seconds

Table 3.1 – Time taken for each configuration in the *Counter flow* test.

D) *Exit flow*: The environment for this test is a large 30x20 meters room with four exits of 1 meter each, positioned along the 30 meters sides at 4 meters from the walls. 1000 agents are uniformly distributed in a 16x26 region at the center of the room. The simulation is done twice, the second time with the exits in one side closed. The IMO's specification for this scenario is illustrated in Figure 3.5. The elapsed time in the second case should be around 50 percent greater than the first to be successful in this test. In our simulations, the first case computed 41 seconds and the second computed 68 seconds, while at a considerable margin, we can consider the test successful.

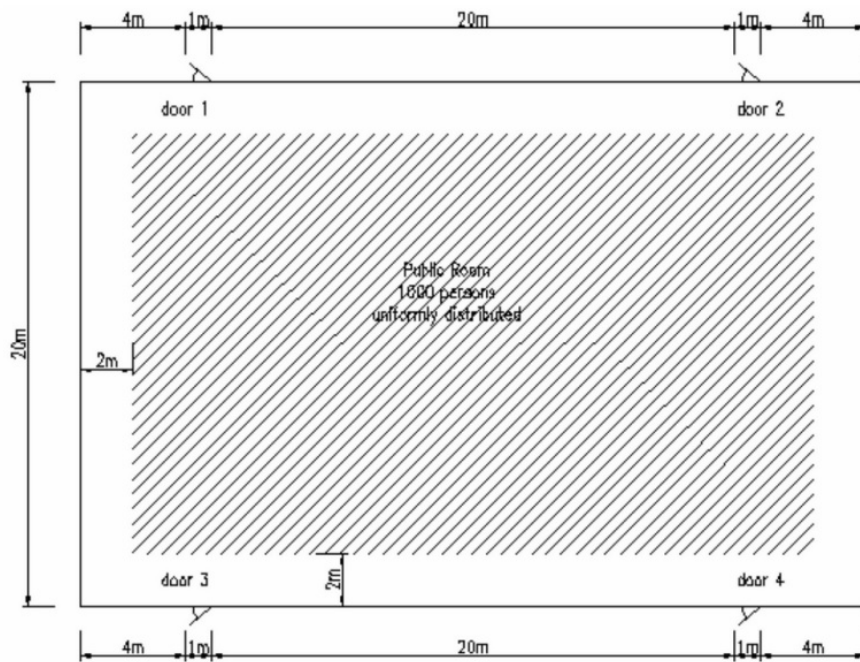


Figure 3.5 – *Exit flow* scenario configuration, following IMO's specification.

E) *Exit route allocation*: This test involves a more complex scenario, a cabin corridor with 12 cabins, two intersecting corridors of different widths, two exits of different sizes and a fixed number of agents according to the cabin. The specification can be seen detailed in Figure 3.6. The criteria for this test is that each agent should move to the closest exit, which means that agents in cabins 1, 2, 3, 4, 7, 8, 9 and 10 should move to the main exit, and the remainder to the secondary exit. In our simulation, the exits were correctly assigned.

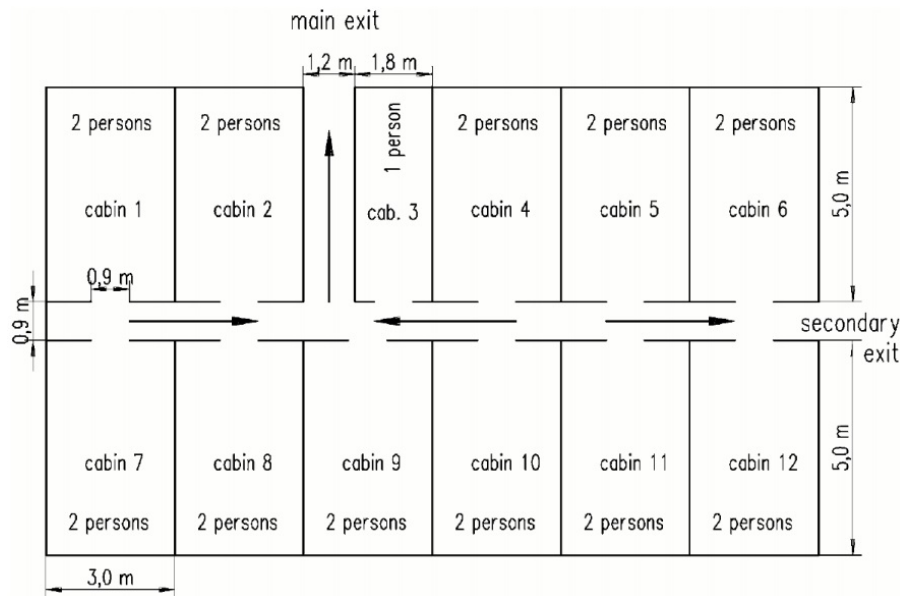


Figure 3.6 – *Exit route allocation* scenario configuration, following IMO's specification.

Considering all performed evaluations comparing our ORCA implementation is the original one and the validations according to IMO, we can say that our simulator is valid. So, in next Section we describe our model to estimate rooms parameters in evacuation scenarios, based on ANNs. Indeed, the neural networks are trained based on data generated using our simulator.

3.2 Estimation of Rooms Parameters based on ANNs

This Section presents our model to estimate crowd data, resulting of evacuation process, instead of simulating it. As said before we are interested about to test ANN as a tool to estimate crowd parameters instead of generate them through simulation. The overview of our method comprehends three phases: Firstly, we performed several simulations in order to compose the dataset to be used to train the ANN (see Section 3.2.1). The second phase is responsible for ANN training and validation, as discussed in Section 3.2.2. The third phase is focused in testing the learning methodology for rooms estimation in complex environments (Section 3.3).

As mentioned before when we explained details about the simulation, the room parameters in our method, as it is designed to compose the dataset to train, validate and test, are following described:

- *Room width* (meters),
- *Room length* (meters),
- *Exit size* (meters),
- *Input flow* (Agents per second that entry in the room),

- *Flow Duration* (Duration in seconds defining the period where agents entry in this room) and
- *Initial population* (Number of agents which are inside the room at the beginning of the simulation).

Then, as in the simulation process, the estimation using ANNs proceeds by estimating the following data for each room i :

- *Evacuation time* tt_i (seconds),
- *Average exit time* \bar{t}_i (seconds),
- *Average speed* \bar{s}_i (meters per second); and
- *Average density* \bar{d}_i (agents per meter).

In addition, other information are computed using our proposed heuristic for the whole environment e :

- *Total evacuation time* tt_e (seconds),
- *Average exit time* \bar{t}_e (seconds),
- *Average speed* \bar{s}_e (meters per second); and
- *Average density* \bar{d}_e (agents per meter).

Details about the estimation procedure is given in next Sections.

3.2.1 Database Creation

To create a database of simulations that could be used to train and test the learning algorithms, we automatized the process of creating rooms. This process generates simulations based on the input parameters, which are randomized for each simulation, and it collects data of the output parameters. Each generated simulation is composed of a square room with a single exit ¹. The room's dimensions vary according to the *Room width* and *Room length* parameters, the exit size vary according to the *Exit size* parameters.

Agents are placed in the room using the *Initial population*, *Input flow* and *Flow Duration* parameters. The agents informed in initial population are the ones whose are already in the room at the simulation beginning, and are created within a diamond pattern at the center of the room to avoid agents overlapping each other. However, as their position is restricted by the size of the room,

¹In this paper we investigate specifically the scope of rectangular rooms and single exits. Different rooms geometry and number of exits are going to be addressed in a future work.

if no space is available to accommodate all of agents that need to be created, it may be inevitable that some agents starts positioned overlapping others. If it happens, when simulation starts, agents will try to avoid collisions among them, eventually finding enough space to move without colliding with other agents.

New agents n are created w.r.t to flow information. These agents are created at the entrance in the opposite direction of the exit in a room i , considering a flow in rate defined as:

$$n_i = f_i \cdot F_i^t, \quad (3.1)$$

where f_i is the *Input flow* defined for room i , and F_i^t is the *Flow Duration* for i at a time t that is included in the interval $[t_i; t_f]$ that represent respectively initial and final time for the people flow in the room. When t is greater than t_f , i.e. the people flow in room i is finished, $F_i^t = 0$, so the entrance of agents in room i stops.

In regard of other parameters present during the simulation: the space each agent occupies is represented through a cylinder with a radius of 0.3 meters and it moves trying to maintain a max speed of 1.2 m/s. Both of these values were extracted from the literature for the average value people size and the average speed people have while walking [43].

ORCA makes use of a prediction time to make agents react and avoid collisions with each other. We were not able to find any research or study about which value must be used to be more close to reality. In fact, the common assumption is that value can vary according to each individual (cultural or personality) and with the environment conditions such as time of day, smoke, fire and crowd density. In our simulations we empirically adopted the value of 3 seconds simply because it was the value which we got better visually convincing collisions avoided during validation tests.

To generate the training and validation database, the room's parameters were randomly defined in the intervals as specified in Table 3.2. We generated 18000 rooms for training and 2000 for validation. The tests are presented later in Section 4.

Parameter	Min value	Max value	Value Type
<i>Room width</i>	2.0	20.0	Float
<i>Room length</i>	2.0	20.0	Float
<i>Exit size</i>	0.9	5.0	Float
<i>Input flow</i>	1.0	10.0	Float
<i>Flow duration</i>	0.2	100.0	Float
<i>Initial population</i>	0	99	Integer

Table 3.2 – Parameters and values.

3.2.2 Training and validation

For each attribute to be estimated (Evacuation time, Average exit time, Average speed and Average density) we trained a different neural network. We started with a simple model of neural network: one hidden layer with six neurons fully connected and no bias. The loss function used was the mean squared error function. The training process only stops when no more reduction in the loss was observed by new epochs even when reducing the learning rate.

With these 4 networks trained, the following individual validation errors were measured using the absolute relative error comparing the predicted value and the simulated value. We measured the quality of our estimations according to how many cases had less than 10% error size.

The Evacuation time estimated by the ANN resulted in 89.5% of the validation cases bellow 10% error, the Average exit time ANN presented 72.3% of the cases, while the Average speed and Average density networks presented 40.9% and 45.7% respectively. The mean errors histograms of these networks can be seen in Figure 3.7.

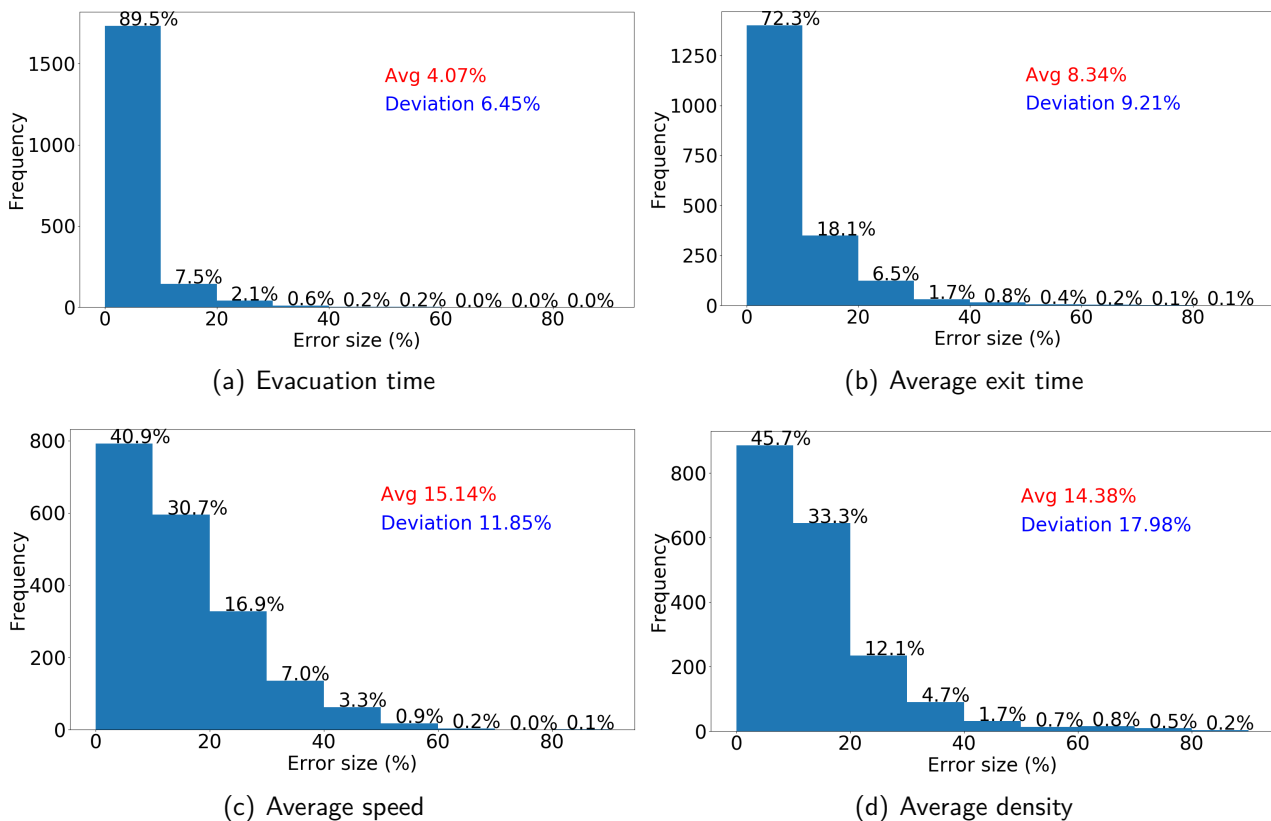


Figure 3.7 – Error frequency of the validation cases with neural networks with one hidden layer with 6 neurons. The number of occurrences of error sizes are grouped in bars of 10% error size, with a percentage indicating how much of the validation cases are in it, also the average error size of all cases and its correspondent standard deviation are shown.

In order to test other ANNs architectures, we perform a variation in the quantity of neurons in the hidden layer and evaluating the results. New attempts were made using 50, 200, 400 and 500

neurons in the hidden layer. Each attempt efficiency was measured by collecting how many cases during individual validation were bellow 10% error size, as it can be seen in Figure 3.8.

The best configuration obtained for each attribute was with 400 neurons for the Simulation time and for Average exit time and 200 neurons for Average density. Average speed had nearly the same result with 200 and 400 neurons, but 200 neurons was chosen because of its slightly smaller standard deviation during individual validation. With these new networks the number of cases bellow 10% error improved for all attributes: from 89.5% to 91.5% in Simulation time, from 72.3% to 85.4% in Average exit time, from 40.9% to 49.9% in Average speed and from 45.7% to 51.6% in Average density. The error histograms can be seen in Figure 3.9. As could be seen, our tests indicate that evacuation and average exit times are parameters that seem to be adequate to be used in such type of ANNs. Fortunately, although the other parameters are also important, these two related to the time are more important in evacuation scenarios evaluation. That is the reason why we pursue our investigation using only these two attributes and keeping densities and velocities for a next work.

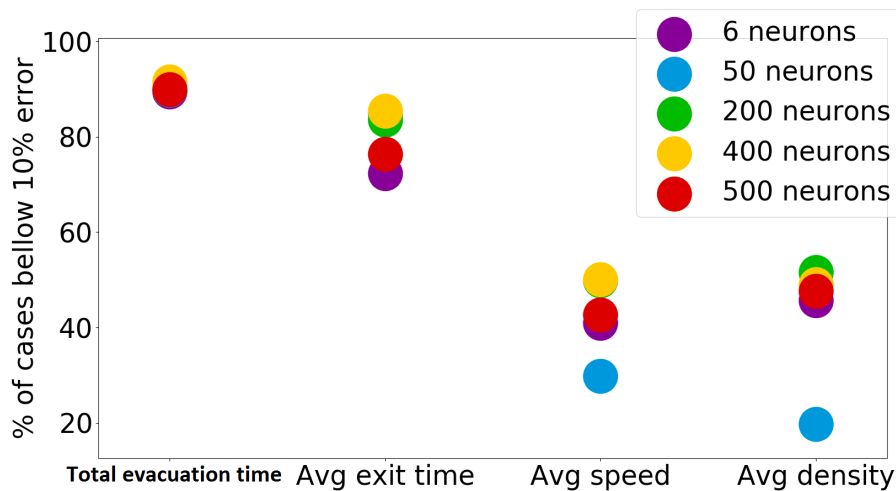


Figure 3.8 – Search for a better network model. Attempts with 6, 50, 200, 400 and 500. Plotted number of cases bellow 10% error in the individual validation.

Next Section describes our heuristic capable of aggregating per-room information in order to estimate complex environments.

3.3 Environment Estimation based on ANNs and Heuristics

As described in last Section, the crowd data to estimate the evacuation times at each room uses ANNs. However, in general, crowd evacuation methods are used for more complex environments than only one room. Therefore, 3D modeling is usually necessary in order to create the environment to be simulated. Once we are proposing a crowd estimation instead of simulation, we propose to create a simple and user friendly tool to help modeling environments based on a graph of rooms (i.e. an environment graph where the connections among the rooms are defined as graph edges).

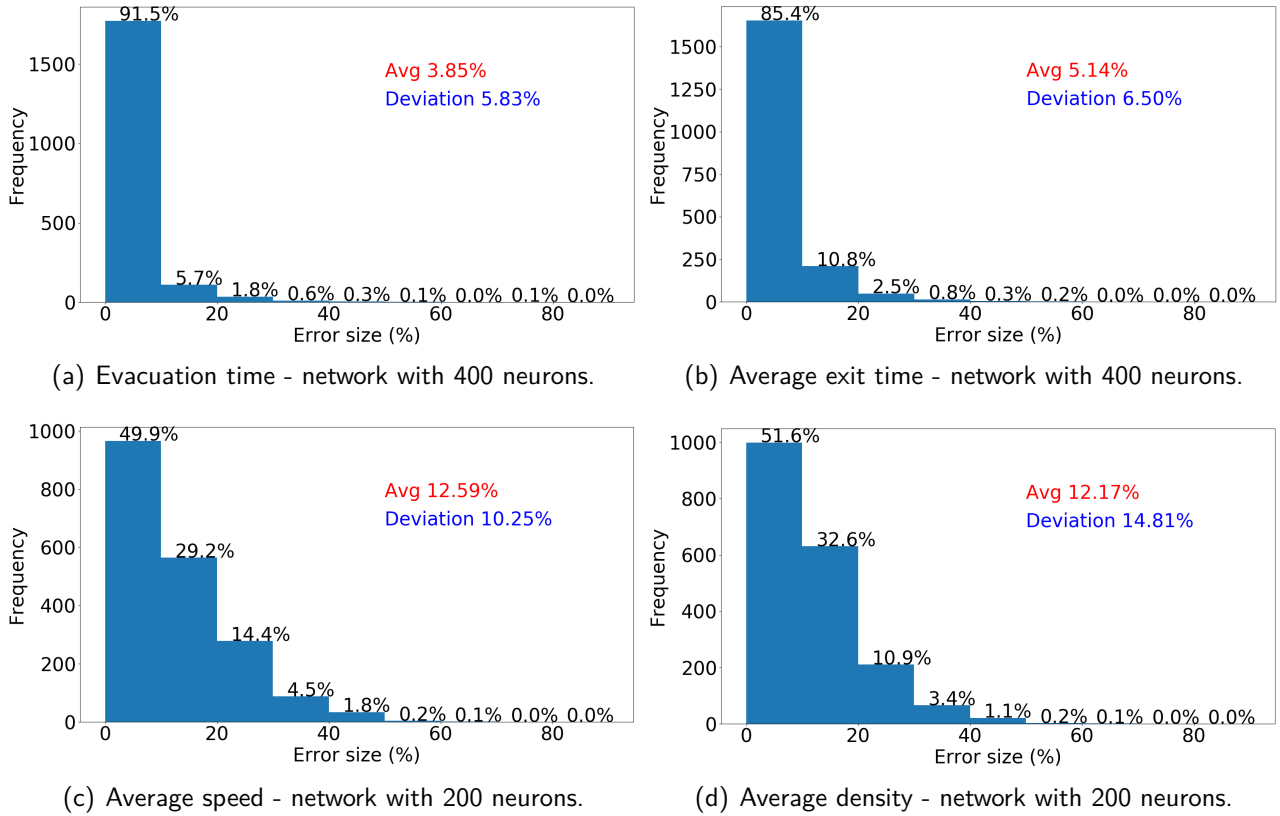


Figure 3.9 – Error frequency of the validation cases with the improved neural networks. The number of occurrences of error sizes are grouped in bars of 10% error size, with a percentage indicating how much of the validation cases are in it, also the average error size of all cases and its correspondent standard deviation are shown.

We developed an environment editor (see Section 3.3.1), where the user can easily create and edit a graph representing an environment to be estimated.

Our proposal is to use ANNs to estimate the parameters from each room (as discussed in last Section) and then combine this data, through an empirically defined heuristic to compute the global data for the whole environment. Heuristics are used in two stages. The first one is responsible for finding the placement of each room in a time-line and calculate the flow of population from one room to other rooms, while considering the hierarchy of rooms in the graph. In the second stage the data regarding the whole environment is computed using the data estimated for each room.

One environment e is composed by N rooms r to be estimated. We estimated the crowd parameters for each r_i using the four trained ANNs, one for each crowd parameter (tt_i , \bar{t}_i , \bar{s}_i and \bar{d}_i), as previously specified. Therefore, we have two types of rooms r in an environment e : i) rooms of type D which population impacts another room, i.e. people from r_{D_i} comes to another r_i and ii) rooms of type E (exit rooms) which are rooms that lead the population to the output of the environment. Moreover, these two types of rooms are exclusive, i.e. rooms D never lead to exit and E rooms never impact any other room in e .

A - The Heuristics during Room Estimations

The environment estimator loads the specified environment and performs the data estimation for each room separately (types D and E) using the ANNs (as described in Section 3.2). As said before, some rooms have dependence rooms, i.e. a certain r_i is impacted by other rooms (type D) which population goes to r_i . This is the definition of the set of r_{D_i} dependence rooms that impact r_i . That is the reason why some rooms have to be estimated before others. So, first we are going to present the execution flow for rooms which do not have dependences ($N_{D_i} = 0$).

1. Since no people enters this room coming from another room: *Input flow* $f_i = 0$ and *Flow Duration* $F_i = 0$,
2. The neural networks are loaded and the two output parameters of the room (tt_i and \bar{t}_i) are estimated using their respective networks. In addition to these estimations, two other features are computed f_{et} and g_{fet} , as follows:

$$f_{et_i} = \begin{cases} \frac{length_i}{maxSpd}, & ip_i = 0 \\ \frac{length_i - Max(0, (\frac{0.6\sqrt{ip_i \cdot (2-1)}}{2}))}{maxSpd}, & otherwise \end{cases} \quad (3.2)$$

where f_{et_i} is the first exit time from r_i which means the time that the first agent exits such room and ip_i is the number of agents initially created in room i . It is simply calculated as the time it takes for an agent to pass through the room at maximum speed in the simulator ($1.2m/s$). However, when $ip_i > 0$, the first agent to exit will not need to travel all the room length to exit since it is placed at the center of the room. To deal with that we designed the second part of Equation 3.2, which represents the distance reduction to be traveled by the first agent to exit the room. The next equation is also computed afterwards the ANN execution:

$$g_{fet_i} = f_{et_i}, \quad (3.3)$$

where g_{fet_i} is the time the first agent exited the room in a global time among all rooms. In the case of rooms without dependence is exactly the value of f_{et_i} since they are the ones that "start" the crowd movement in the estimation. Therefore, f_{et_i} and g_{fet_i} are used to propagate values to the input of rooms that are their dependents.

For each room r_i which $N_{D_i} > 0$ we perform the following execution flow:

1. If the room has a dependence (r_{D_i}) that is not estimated yet, the estimation of this room is postponed until all dependences are estimated. If all (r_{D_i}) are already estimated then we compute the following data for r_i :

$$git_i = \min(gfet_{D_i}), \quad (3.4)$$

$$ift_i = \max(gfet_{D_i} - fet_{D_i} + tt_{D_i}), \quad (3.5)$$

where git_i is the global initial time of r_i , i.e. the time this room receives its first agent coming from a r_{D_i} in a global time among the rooms, and ift_i is the income final time of r_i , i.e. the global time the last agent coming from a r_{D_i} enters into r_i . These two variables together determines a window in time in which the dependence rooms of r_i will send agents to r_i , as described in Equation 3.6. The input flow is determined as Equation 3.7. This way all agents coming into r_i from its dependences are considered a constant flux of agents during the time window.

$$F_i = ift_i - git_i. \quad (3.6)$$

$$f_i = \frac{\sum_{k=1}^{N_{D_i}} (pop_k \cdot p_{k,i})}{F_i}, \quad (3.7)$$

where N_{D_i} is the number of dependence rooms of i and $p_{k,i}$ is the percentage of agents that moves from the dependence room k of i to room i . pop_i is the population of r_i , and it is computed as follows:

$$pop_i = ip_i + (f_i \cdot F_i), \quad (3.8)$$

where ip_i is the number of agents initially created in room i .

2. With all the data needed from dependences propagated in room i , the neural networks can estimate the output parameters of such room (tt_i and \bar{t}_i), just as before.

The parameter $gfet_i$ is then calculated as follows:

$$gfet_i = fet_i + git_i. \quad (3.9)$$

Equation 3.9 is different from 3.3 because in the first case room i has $N_{D_i} <> 0$, i.e. this room has dependences. Consequently, agents pass arrive this room coming from another one, so the time it takes to the first one to exit has an offset of the time it took for someone to enter r_i . In the second case, $N_{D_i} = 0$, agents start to be simulated in time = 0.

After we finish the estimation of a certain r_i this whole process is repeated until all N rooms in the environment are estimated. It is important to emphasize that these simple equations have the only goal to aggregate the flow of population passing through the rooms and the global times. Other equations to aggregate could also be possible, we tested some but at the end these bunch of equations had the better results. This is certainly a possible future work for this research.

B - Final Data Estimation Regarding the Environment

The last step of our method is to estimate the crowd parameters for the whole environment based on the rooms estimation. We first consider the exit rooms data (type E) to estimate the *Evacuation total time* tt_e of a specific environment e . Indeed, this is the greatest Total evacuation time of M existent exit rooms that represent the exits in the environment, computed as Equation 3.10:

$$tt_e = \max(\text{git}_{E_e} + tt_{E_e}), \quad (3.10)$$

where E_e are the exit rooms of the environment e . The *Average exit time* \bar{t}_e of the environment is the average between the M exit rooms exit time plus the average time agents enter this exit room, as Equation 3.11:

$$\bar{t}_e = \frac{\sum_{er=1}^M (\bar{t}_{E_{er}} + \frac{\text{git}_{er} + \text{ift}_{er}}{2})}{M}. \quad (3.11)$$

For the *Average speed* of the environment \bar{s}_e we use average of the *Average speed* estimated for each r_i :

$$\bar{s}_e = \frac{\sum_{i=1}^N \bar{s}_i}{N}. \quad (3.12)$$

Similarly we compute the *Average density* of the environment \bar{d}_e average of the *Average density* obtained for each r_i :

$$\bar{d}_e = \frac{\sum_{i=1}^N \bar{d}_i}{N}. \quad (3.13)$$

This is the vector 4d containing estimated data for a certain environment e we called $E\vec{D}_e = \{tt_e, \bar{t}_e, \bar{s}_e, \bar{d}_e\}$. We used $E\vec{D}_e$ to compare with $S\vec{D}_e$, resulting from the simulation process, as discussed in Section 4.

3.3.1 Environment editor

The Environment editor is a simple graphical application to allow users to easily create and edit environments to be used for estimation. It consists of a main window divided in two regions, the control panel and the canvas, as shown in Figure 3.10. The control panel provides a file menu

with options to save the current environment into a file, load an environment from a file and to clear the current environment. Also it has a list of available tools and a button to add a new room to the current environment.

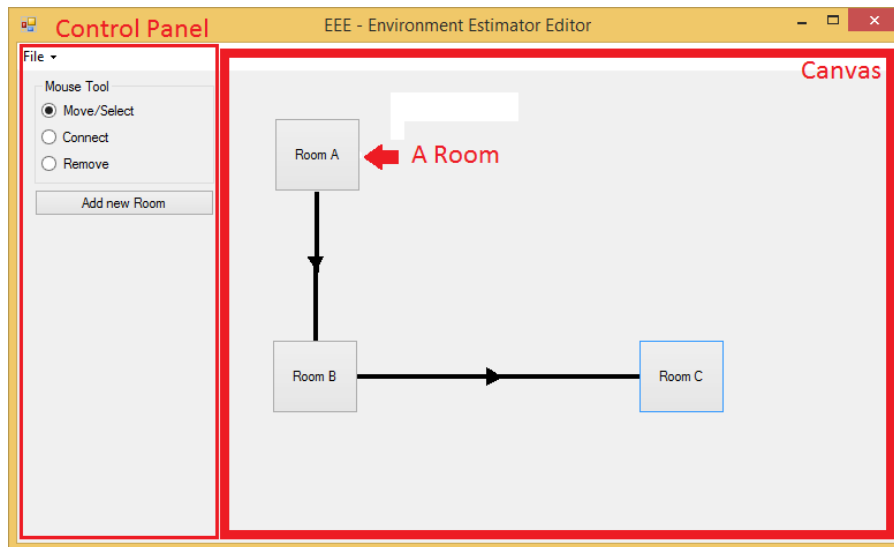
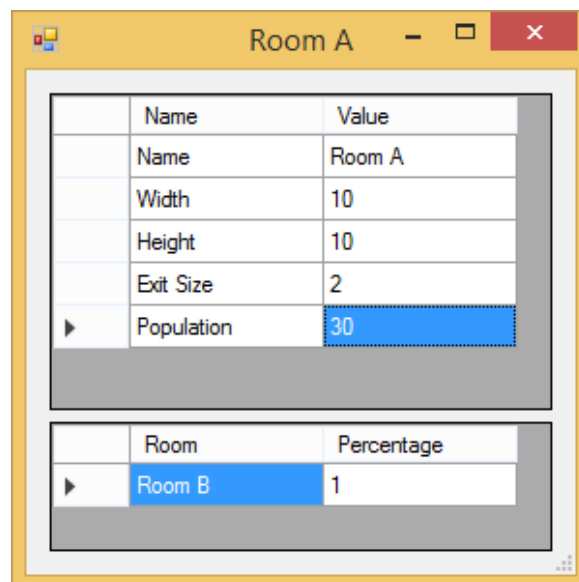


Figure 3.10 – Environment editor main window, control panel and canvas regions and a room are highlighted in red, black arrows indicate each rooms connections to the others.

The tools available can be used to: *i)* move a room in the canvas in order to locate and organize it as the user wants; *ii)* create a connection from one room to another, indicating that the agents from the first room moves towards the second one; *iii)* remove an existing room, together with its connections.

It is also possible to select a room and check its parameters. In this case, a new window will appear, as in Figure 3.11, showing the *Room width*, *Room length*, *Exit size* and *Initial population* parameters along with the connections presented in the selected room. The value of the parameters can be modified by the user and the connections may be removed.



	Name	Value
	Name	Room A
	Width	10
	Height	10
	Exit Size	2
▶	Population	30

	Room	Percentage
▶	Room B	1

Figure 3.11 – Environment editor's room values window, it shows the values of the parameters of the selected room for the user to modify it to design the environment.

4. EXPERIMENTAL RESULTS

This Section describes some results we obtained testing our method. Section 4.1 describes experimental results when simulating and estimating 20 full scenarios with varied population distribution in the rooms. In Section 4.2 we modeled 28 new environments by replicating rooms and evaluated them in order to measure the impact of ANN error. Finally, we proceed with a practical usage of our method applying it in a real life environment, as described in Section 4.3.

4.1 Evaluating Environments with Varied Populations

For the first set of scenarios to be tested, we created a quantity of full varied environments to be estimated and simulated. Our goal is to analyze the obtained errors when comparing our method for estimation with the numbers obtained with the full simulation.

Using the Environment Editor mentioned in Section 3.3.1 we have manually created 10 scenarios to be used with two environment cases created for each of these scenarios: one with all agents placed at the first rooms of the scenario i.e. the ones with no dependencies, and other with the agents placed as evenly as possible among all rooms in the scenario. The scenarios created can be described as:

- Sim 1: This scenario has 4 rooms. The first room is 10 meters wide and 5 meters long, it bifurcates in two smaller rooms of 4x5 meters. These both rooms connect to the same room, another 10x5 meter room which leads to the exit. The rooms' doors that lead to other rooms are 1 meter wide, while the door to the exit is 2 meters wide. 20 agents are present in this scenario.
- Sim 2: Three 5x5 meters rooms are present in this scenario. They are sequentially positioned, with the first room leading to the second, the second leading to the third and the third to the exit. The door size of each room increases incrementally, the one from the first to the second room being 1 meter wide, from the second to the third 1.5m and the third to the exit 2m. The number of agents in this scenario is also 20.
- Sim 3: Another scenario with 4 rooms. This time 3 rooms of different width leads to a bigger room that leads to the exit. The dimensions of these rooms are 5x5, 2x5 and 3x5 meters for the first three rooms and 11x5 meters for the bigger room. All doors have 2m size. 30 Agents are present.
- Sim 4: This scenario is similar to Sim2, but with 4 rooms instead of 3. The rooms have also 5x5 meters and are sequentially positioned. The doors also have incremental size, from 1 to 2.5 meters. This time and the number of agents present is 30.

- Sim 5: Five rooms also sequentially positioned are present here. The first, third and fifth rooms are 5x5 meters. The second and fourth rooms act as a corridor between them, with 2x12 and 2x7 meters, respectively. All doors are 1m wide, including the one to the exit. 20 agents are present in this scenario.
- Sim 6: One more scenario with 4 rooms sequentially positioned. This time the rooms are smaller 2x3 meters rooms, except for the fourth room which has 2x4 meters. All doors have 1 meter size and 28 agents are present.
- Sim 7: Also similar to Sim2, but with five 5x5 meters rooms also sequentially positioned instead of three. The doors also are not exactly equal, decreasing in size instead of increasing, from 3 meters, in the door in the first room to the second, to 1 meter in the door from the last room to the exit.
- Sim 8: Another similar scenario to Sim2. Five 5x5 meters rooms sequentially positioned are present instead of three. This time the doors have not incremental sizes, all doors have 2 meters size. 20 agents are also present in this scenario.
- Sim 9: Very similar scenario to Sim8, but with seven rooms instead of five. The dimensions of the rooms, size of doors and number of agents remains unchanged, being 5x5 meters rooms with 2 meters large doors and with 20 agents present.
- Sim 10: Also similar to Sim8, but with nine rooms instead of five. rooms have also 5x5 meters, doors have also 2 meters and also 20 agents are present.

An example of one of these environments modeled in the Environment Editor can be seen in Figure 4.1 where one case with the Sim 8 scenario is modeled.

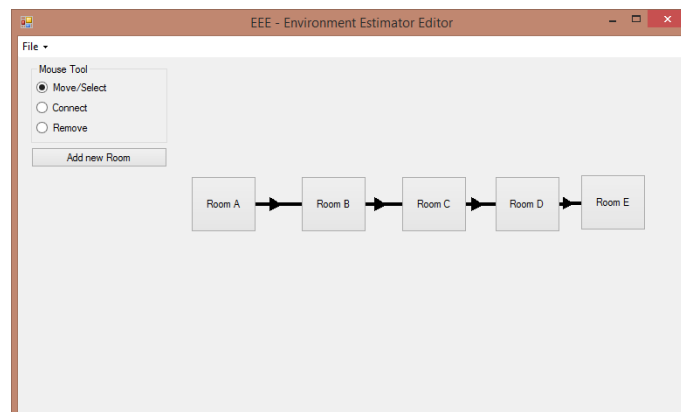


Figure 4.1 – A environment with the scenario Sim 8 modeled in the environment editor.

In order to simulate these environments, to perform the comparison between estimated and simulated values, we have modeled the 3D geometry and simulated them using the simulator we developed in Unity. The modeled environments in this task can be seen in Figure 4.2. For each

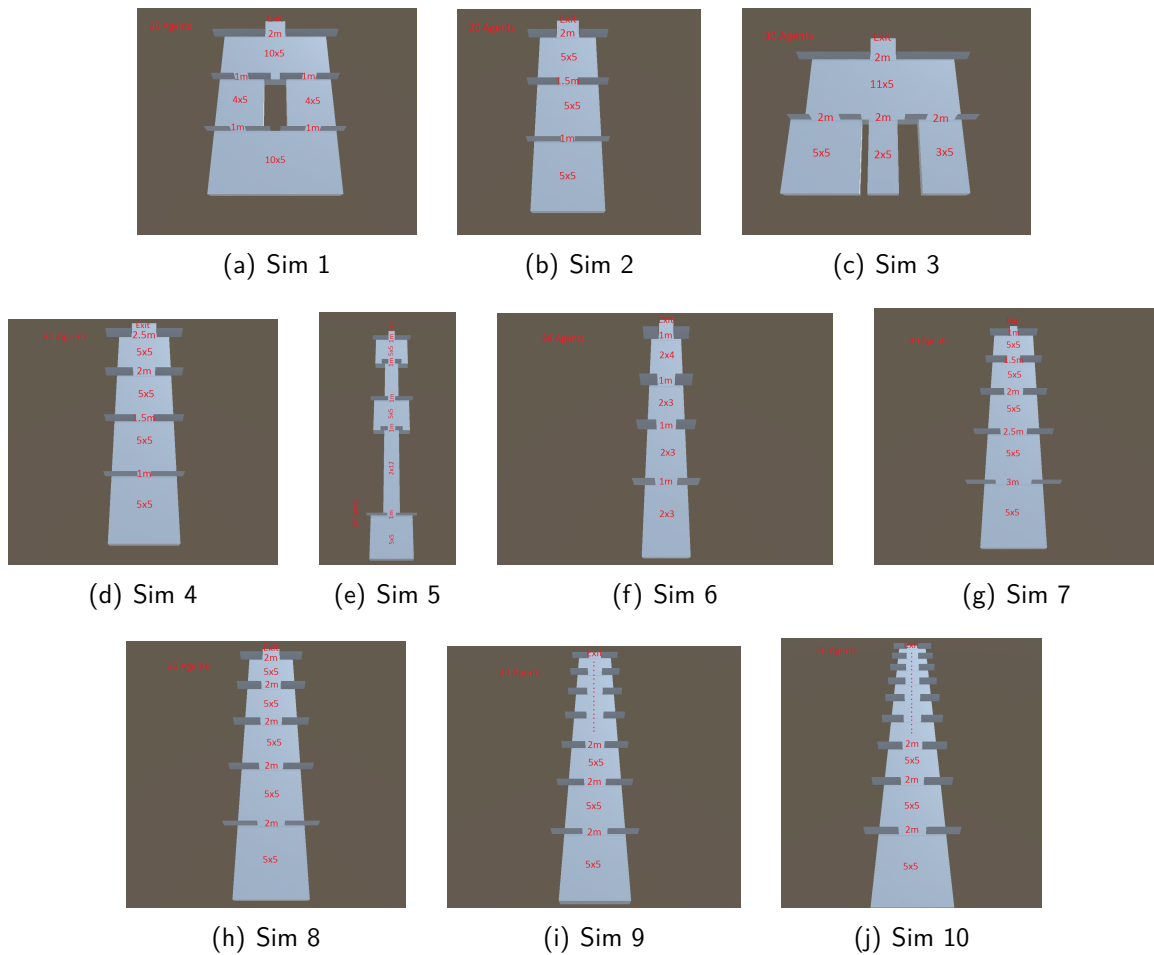


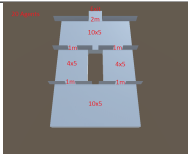
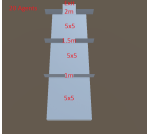
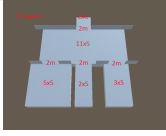
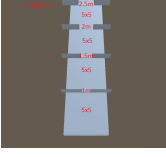
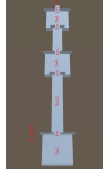
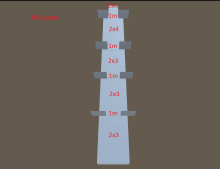
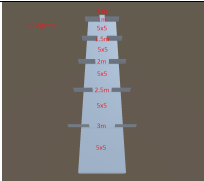
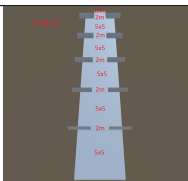
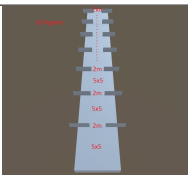
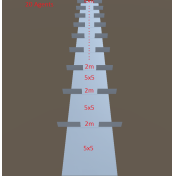
Figure 4.2 – Scenarios modeled in the Simulator developed in Unity.

environment we simulated two cases, as previously mentioned: i) with all agents in the first room in the hierarchy and ii) with same number of agents distributed along the rooms.

The results of the comparison for the first environment cases, when the agents are placed in the first rooms of the scenario, are shown in Table 4.1, in the 3th and 4th columns. For the second set of cases, where agents are distributed evenly on the rooms of the scenario, the results are in the 5th and 6th columns. These columns contains respectively the percentage relative error of total evacuation time (tt) and average time per agent (\bar{t}) when compared to the values achieved in the simulations.

Figure 4.3 shows the measured error values for all 20 environment cases regarding total evacuation time tt_e and average time \bar{t}_e . In this Figure it is easy to see that when the population is distributed along the rooms ($Sim1'$ to $Sim10'$), the error value of average time \bar{t} for each agent increases. It is expected in our method because each room is estimated only once for each full environment. It means that when the population is distributed in various rooms, people first wait for agents that should entry in a specific room in order to be estimated. For example, consider $Sim6'$ as illustrated in Table 4.1: if there are agents in the central room, those entities should wait for others that come from the other rooms in order to have the central room estimated by the ANNs.

Table 4.1 – Environment cases configuration and estimation relative errors.

Scenario	Model	tt first rooms	\hat{t} first rooms	tt agents distributed	\hat{t} agents distributed
Sim 1		-29%	-15%	-2%	25%
Sim 2		-31%	-17%	-5%	40%
Sim 3		7%	18%	16%	23%
Sim 4		-25%	-7%	-21%	35%
Sim 5		-3%	1%	-2%	45%
Sim 6		36%	40%	-15%	18%
Sim 7		-10%	-1%	-25%	-7%
Sim 8		26%	16%	14%	62%
Sim 9		22%	13%	13%	54%
Sim 10		20%	12%	13%	50%

This characteristic impacts mainly the parameters \bar{t} , since agents that are initially placed just a room away from the exit should wait until all other agents are in this specific room for it to be estimated.

The obtained average errors for total evacuation time tt for scenarios Sim1 to Sim10 is 20.9%, while the average error for scenarios Sim1' to Sim10' is 12.6%. It can be interpreted that our method estimates reasonably well the total evacuation time for full environments, even if the population distribution varies. Reminding that, as previously discussed, the limitations of our heuristic impacts mainly in the average time \bar{t} parameter. With that said, the error of \bar{t} for the first set of simulations is 14% while the second set presented an error value of 35.9%. This difference in the error obtained for these two different types of environments confirms the expected side effects in the estimation. In addition, the tt average error value for the 20 environments is 16.75% and the \bar{t} average error value is 24.95%.

Percentage Errors observed in Estimated Parameters

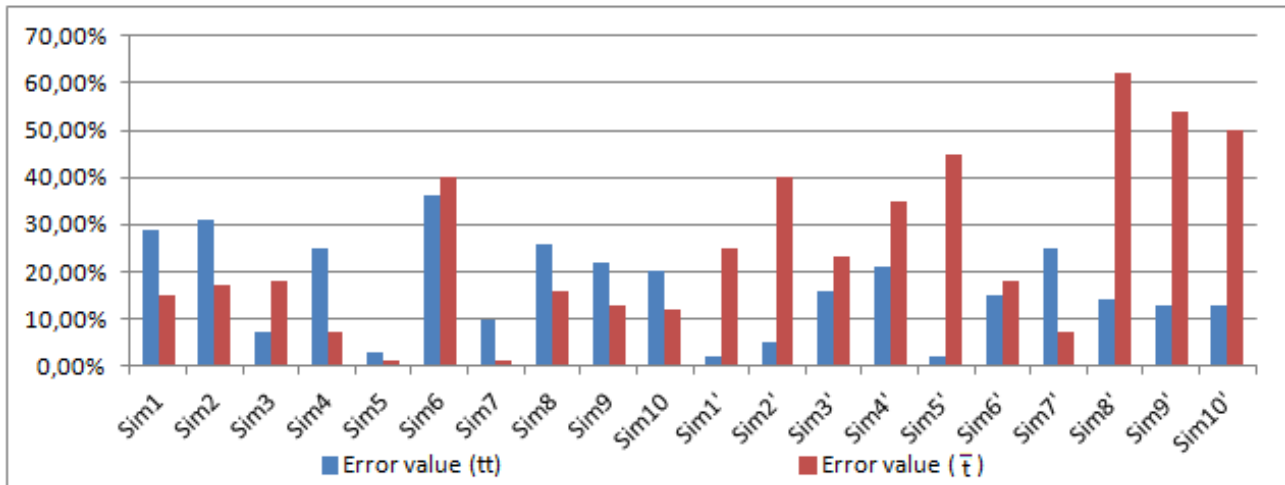


Figure 4.3 – Obtained results when testing 20 environments. The percentage of errors are shown for total evacuation time tt_e and for average time \bar{t}_e metrics for each environment.

4.2 Investigating Replicated Rooms

We expect that the estimation using ANNs will present errors when compared to full simulation. To expect them to predict perfectly the behavior of the crowds would be too optimistic. As the heuristics fill data for rooms using information obtained from the ANNs, these errors can be propagated forward in the graph, possibly leading to even bigger errors.

In order to evaluate this error propagation behavior, we handpicked 2 rooms from the 2000 ones used in validation. We selected the two rooms with the best and worst estimations of the total evacuation time (tt) presented in this set of environments. They are following named: r_b and r_w , respectively for the best and worst estimated total evacuation times. r_b is a 28 meters wide 6 meters

long room with 99 initial agents and an incoming flux 3.3 agents per second during 24.7 seconds with an exit 5.6 meters wide, while r_w has 29.5x4.7 meters, 12 initial agents and 9.3 agents per second during 1.4 seconds and a 1.1 meters wide exit.

For each of those rooms, we generated new full environments by replicating the specific room structure (dimensions and exit size for r_b or r_w) and positioning them sequentially. The agents are placed in the first room of the sequence and must travel to the exit, in the last room of the sequence. Figure 4.4 illustrates one example of environment with replicated rooms, having 9 rooms.

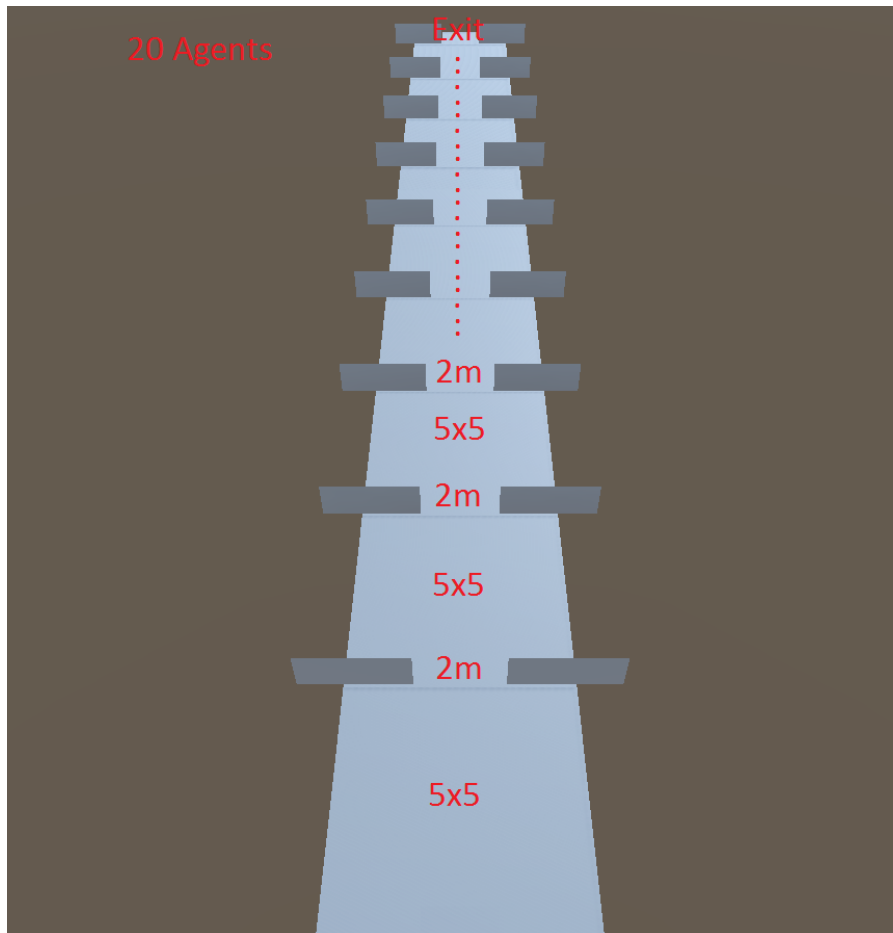


Figure 4.4 – Nine identical rooms placed in sequence in order to be estimated.

Using this method we created 14 new environment cases for each of the selected rooms, each with two more rooms than the last one. For instance, the first created environment has 3 rooms, then the second environment has 5 rooms, then 7 rooms, 9 rooms, until 29 rooms. In total we generated 28 new environment cases, half of them replicating r_b and the other half r_w . In all cases the agents are placed in the first room and must travel until the exit, i.e. the last room.

For each one of these 28 environments, in addition to the graph representation using our editor (see Section 3.3.1), we also modeled the geometry using Unity. The goal here is the same of the last experiment, to simulate the environments in order to compare with the estimation and analyze the results, that can be seen in Figure 4.5. In the case of the best estimated room r_b the

average absolute relative error tt in all 14 environments is 18% (Std.Dev=6%) while using r_w is 29% (Std.Dev=15%).

These environments containing 29 rooms in a sequence can be considered a complex scenario and maybe even unrealistic, and although the environments with r_w achieved high error rates, as the number of rooms increases both r_w and r_b converged its error value between 15% and 20%. We consider this as a good result since the error percentage did not increased as the number of rooms increased. It indicates that the propagation of the ANNs errors by the heuristics did not produced an ever growing error rate as the number of rooms increase.

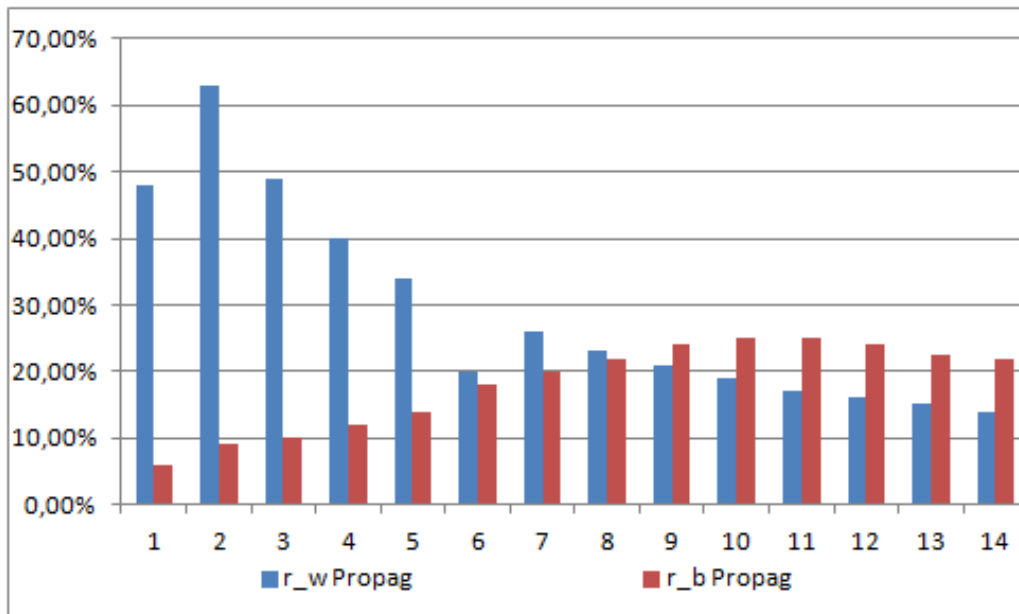


Figure 4.5 – Absolute relative evacuation time (tt) error sizes for each of the propagation cases, ordered by the number of rooms (14 cases, starting with 3 rooms and ending with 29 rooms).

4.3 Testing a practical example

In order to test our method in a practical example, we use the night club as described in [7]. We used the same 3D environment as illustrated in Figure 4.6 to define the environment to be estimated using our editor and simulate it containing 940 agents.

The 3D scenario of the night club has 3 floors and many rooms. Some of these rooms are large, have a more complex shape and have two exits apart from each other. We divided those rooms in two, splitting the population present on it in a proportional way to the area each room has. Indeed, this can be interpreted as if the agents used the exit closest to them in the room. Clearly, this approach is not a perfect representation of the environment, however as there is a lot of the scenario already being abstracted during the estimation, the intent of this method is to reduce the difference between the estimated environment and the simulate one. Figure 4.7 shows the graph generated using our Editor to model the night club (named SM) in order to be estimated.

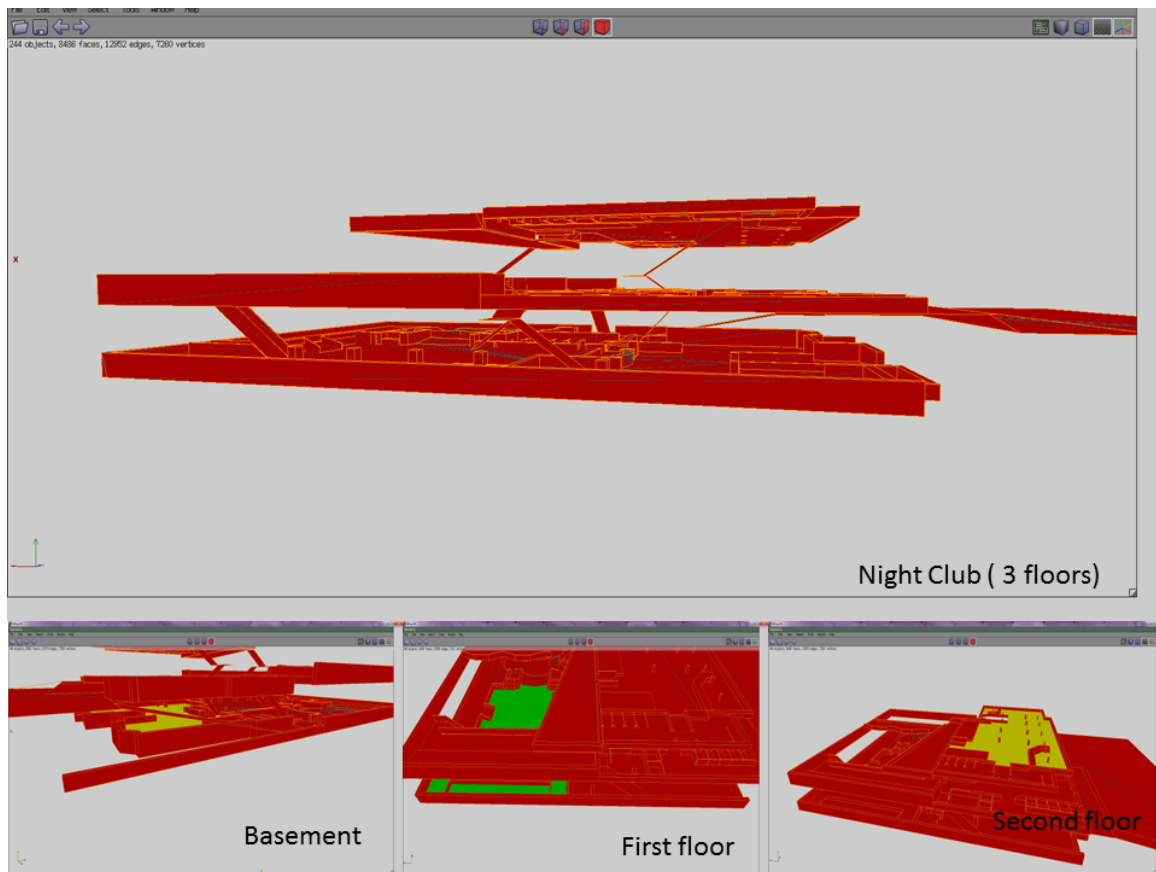


Figure 4.6 – Architecture of the nightclub used as base for simulation and estimation.

For this practical result, we decided to measure the four metrics initially studied in this work (total evacuation time, average time, average speed and average density). The computed differences are presented in Figure 4.8. As can be seen, the total evacuation time keeps an acceptable error around 30%, as the average speed and density. On the other hand, the average exit time per agent presents a larger error mainly due to the fact that the agents are placed distributed in all rooms inside the nightclub, running into the distribution problem already mentioned in Section 4.1.

In terms of required time to model and simulate environments as well as to estimate, follows a comparable analysis:

- **5-10 minutes** was the time to model the night club in our Editor, once the user has the floor plan with the detailed information needed of the environment;
- **12-16 hours** is the time for a skillful designer to model the 3D environment, once s/he has the floor plan of the environment. Additionally, the time to prepare a 3D geometry to be simulated, using a crowd simulator is a complex task. The user has to learn how to operate a simulation software, define rooms to walk, place people and groups, define escape routes. The time to accomplish this task, using CrowdSim as defined in [7], to prepare the night club to be estimate can take **weeks**, and this is repeated for each new environment to be simulated.
- The environment estimation is instantaneous, i.e. to estimate the night club it takes **less than one second**.

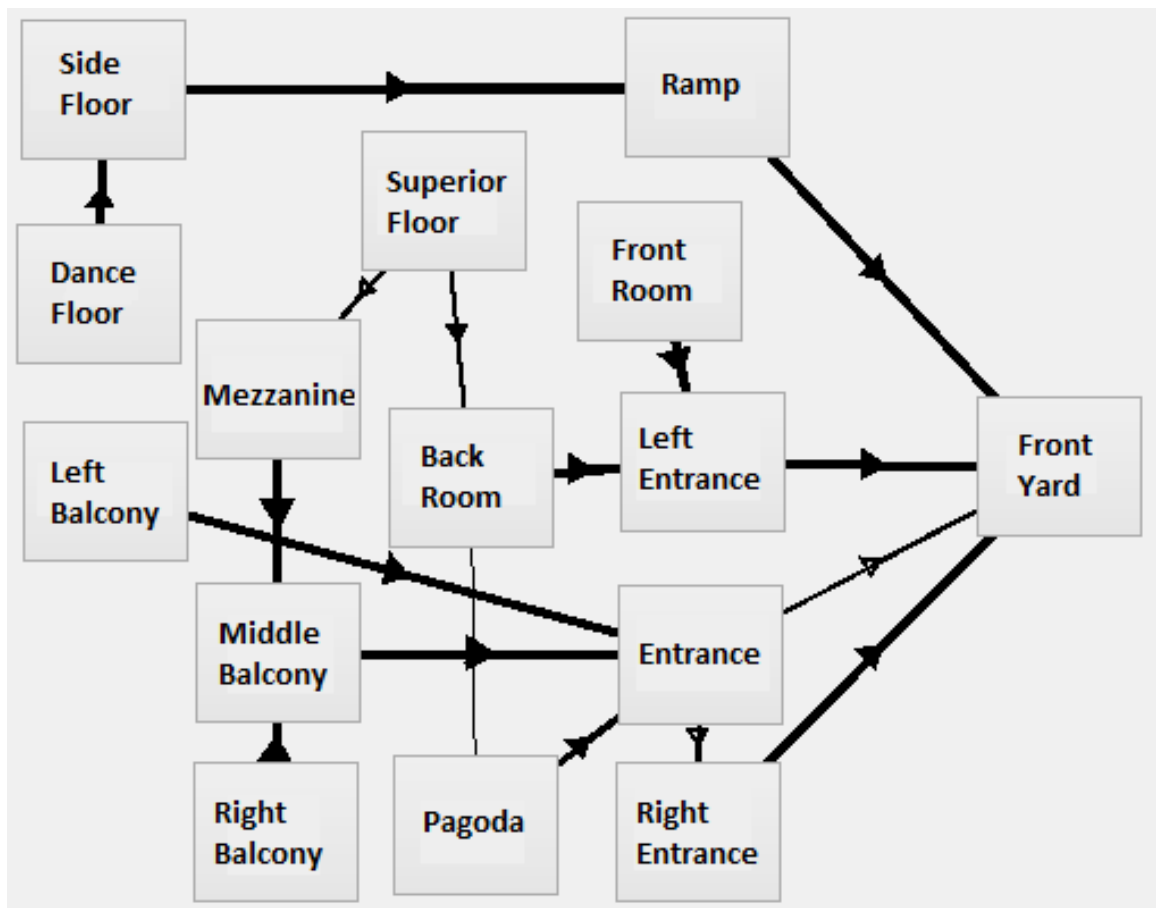


Figure 4.7 – Modeling of SM night club in the Environment Editor.

- A full simulation containing 240 agents, as described in [7] executes at 24 frames/second being the total evacuation time around **3 minutes** for 240 people in the night club.

In the work of Cassol et al. [7], mentioned earlier, authors propose a method to find out the best evacuation plan from several ones, where the difference is the distribution of population among the possible paths to exits. The configuration which had the best result according to their proposed metrics is elected as the best plan. However, this search is severely limited due to the computational power necessary to simulate all the possible distributions a single plan could generate, for this reason they assumed a granularity of 10% to alter these configurations, i.e. in each bifurcation, population varies with a granularity of 10%, for instance 100% and 0%, 90% and 10%, 80% and 20% and etc. An example investigated in the paper mentioned the night club (the same one we used in Section 4) with 240 agents and 3 bifurcations it led to the generation of 1331 plans with altered configurations (distribution in the bifurcation), taking around 8 hours for all to be simulated. Another example, 1010 agents with the same scenario, took around 60 hours, simply because of the additional processing required.

With our estimations replacing the simulations, not only the required time to find the plan is reduced, but more details could be explored as the granularity limitation could be reduced or even removed. More importantly, more bifurcations can be defined in the plan without the worry of taking

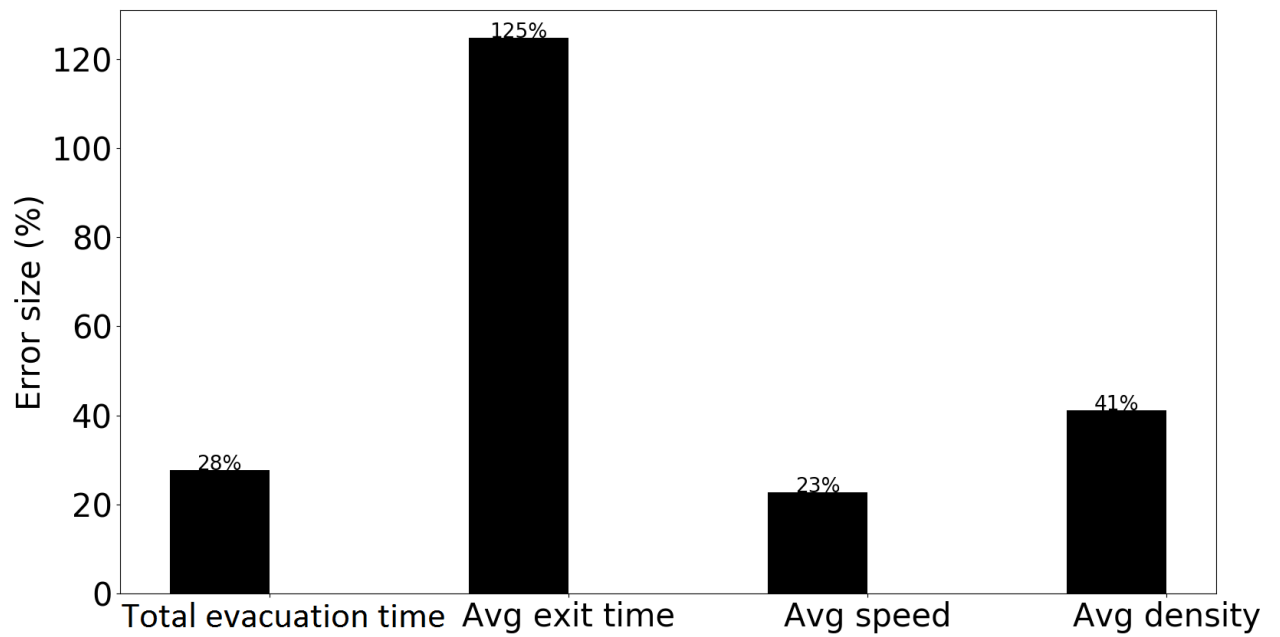


Figure 4.8 – Errors in percentage as resulted from the comparison between the estimation and simulation of a practical example.

too much time to process, allowing for more diverse searches. This replacement would generate a lot of benefits, having only an acceptable estimation error of 30%.

5. FINAL REMARKS

In this document our main contribution is a proposed methodology for estimating evacuation data in simple or complex environments, which is meant to be used for obtaining evacuation data instead of simulating a crowd in the environment. The reasons to this approach is to optimize resources and time since model 3D environment, configure simulations and finally simulate cases scenarios require much more time than the estimation process, as recently discussed in last section. However, the proposed optimization just makes sense if the achieved error is acceptable. For this, we compared the estimated results with simulated ones and measured the accuracy of the different steps of the estimation. Additionally, we investigated potential cases which could be problematic and analyzed results.

In this work, we used ANNs to train and estimate data for separated rooms individually, considering their geometry and population. Then we defined an heuristic to estimate the full environment based on these individual estimations. This approach seems promising as we achieve less than 5% average error when comparing the total evacuation time from the estimation and simulation results for 2000 individual rooms and less than 25% average error in 50 full environments, where complexity was reasonable.

An alternative approach to solve the problem of this dissertation was to train the ANNs considering the full environments instead of splitting them in a set of rooms and estimating them individually. The problem would be how to modularize it in a way to be able to estimate different environments without needing to design and train new neural networks for each new case needed.

About the performed investigations, we have detected a limitation imposed by our approach which happens specifically for rooms which have dependence rooms and also have an initial population on it. Since the ANNs estimation of these rooms could only be executed when the rooms it depends have already been estimated, it generates a conceptual issue where it can be interpreted as if people in a specific room only start to move when people from dependence rooms arrive. We measured this issue in the tests presented in Figure 4.3 in Section 4.1 and, as it could be seen, this impacts clearly the average time the agents took to evacuate the environments. However, on the positive side, this did not impact significantly the total evacuation time of the environments. We intend to improve how the estimation and heuristic occur in our method in order to eliminate this issue.

Other investigation analyzed the suspicion that the errors of the individual estimation of rooms can be carried from a room into another through the heuristics, accumulating and increasing as the environments grow in complexity. We measured the errors produced in Figure 4.5 in Section 4.2 and we concluded that this accumulation does not occur. In fact, the results obtained indicate that this propagation may even help mitigate the error proportioned from a possible bad estimation of a room along the environment, as the error decreases as the complexity increases in the worst case scenarios.

In addition to these investigation results there is still room for improvements in our model. The neural network estimations can be further explored using deeper network models and by increasing the database (currently we have 18000 rooms in training database). These improvements can reduce the error size of the results obtained, specially for the Average speed and Average density since they are the ones with lower accuracy in the individual room estimation.

Also, the training database was created solely with abstracted rooms, containing only one entrance and one exit, always directly opposed to the entrance. By loosening this abstraction we could add rooms with two or more entrances or exits, placing them in the far right or left in the rectangular room.

Still in the abstracted issue, the room did not consider the most common of obstacles like tables, chairs, pillars, couches and other common mobiles in general. One way to improve the versatility of the model and to approach its usability for real cases is to add these informations into the rooms, however as this data is complex, with size and positioning for each obstacle, we suggest to simplify it into an "obstruction level" variable which would indicate how much of the room available to the agents to walk.

Finally, the test performed in a practical example (using one environment which exists in real life) was very promising. The estimation error of total evacuation time was less than 30%, which nicely compensates the facts that the user do not need to model the 3D environment, learn how to use and prepare the simulation in a complex crowd simulator, test simulations and finally wait minutes or hours for results. As described in Section 4.3, this task can save weeks (maybe months) of work. We believe our tool is a further step in the direction to have all environments from real life simulated and tested, since this tool can clearly facilitate this work.

Additionally, if we want to use our method to search for an optimized evacuation plan, similar to what Cassol et al. proposed [7], as mentioned earlier, we believe that the accuracy error obtained would little affect the final result. The obtained error rate is consistent enough so that it could still compare the efficiency of each plan correctly even if the correct value is not exactly achieved. If precision is a necessity, once the best plan is found using our estimation, we can simulate just this best plan and then to obtain the accurate values of the evacuation, saving the time of several in-between simulations that would be necessary otherwise.

BIBLIOGRAPHY

- [1] Aik, L. E.; Choon, T. W. "Simulating evacuations with obstacles using a modified dynamic cellular automata mode", *Journal of Applied Mathematics*, vol. 2012, April 2012, pp. 17.
- [2] Arkin, R. C. "Path planning for a vision-based autonomous robot". In: *Proceedings of the Cambridge Symposium on Intelligent Robotics Systems*, 1987, pp. 240–251.
- [3] Cai, Z.; Yu, Z. L.; Liu, H.; Zhang, K. "Counting people in crowded scenes by video analyzing". In: *Proceedings of the IEEE 9th Conference on Industrial Electronics and Applications*, 2014, pp. 1841–1845.
- [4] Canter, D. V. "Fires and human behaviour". John Wiley & Sons, 1980, 338p.
- [5] Cassol, V.; Dal Bianco, C. M.; Carvalho, A.; Brasil, J.; Monteiro, M.; Musse, S. R. "An experience-based approach to simulate virtual crowd behaviors under the influence of alcohol". In: *Proceedings of the 15th International Conference on Intelligent Virtual Agents*, 2015, pp. 124–127.
- [6] Cassol, V.; Oliveira, J.; Musse, S. R.; Badler, N. "Analyzing egress accuracy through the study of virtual and real crowds". In: *Proceedings of the IEEE Virtual Humans and Crowds for Immersive Environments*, 2016, pp. 1–6.
- [7] Cassol, V.; Testa, E.; Jung, C.; Usman, M.; Faloutsos, P.; Berseth, G.; Kapadia, M.; Badler, N.; Musse, S. R. "Evaluating and optimizing evacuation plans for crowd egress", *IEEE computer graphics and applications*, vol. 37.4, July 2017, pp. 60–71.
- [8] Cassol, V. J.; Rodrigues, R. A.; Carneiro, L. C. C.; Silva, A.; Musse, S. R. "Crowdsim: Uma ferramenta desenvolvida para simulação de multidões". In: *Proceedings of the 11th Brazilian Symposium on Computer Games and Digital Entertainment, SBGames*, 2012, pp. 4.
- [9] Chan, A.; Vasconcelos, N. "Bayesian poisson regression for crowd counting". In: *Proceedings of the IEEE 12th International Conference on Computer Vision*, 2009, pp. 545–551.
- [10] Chan, A. B.; Liang, Z.-S. J.; Vasconcelos, N. "Privacy preserving crowd monitoring: Counting people without people models or tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–7.
- [11] Chu, M. L.; Parigi, P.; Law, K.; Latombe, J.-C. "Modeling social behaviors in an evacuation simulator", *Computer Animation and Virtual Worlds*, vol. 25–3-4, May 2014, pp. 373–382.
- [12] de Lima Bicho, A. "Da modelagem de plantas à dinâmica de multidões: um modelo de animação comportamental bio-inspirado", Ph.D. Thesis, Universidade Estadual de Campinas, 2009, 114p.

- [13] Fradi, H.; Dugelay, J.-L. "Crowd density map estimation based on feature tracks". In: Proceeding of the IEEE 15th International Workshop on Multimedia Signal Processing, 2013, pp. 40–45.
- [14] Fruin, J. J. "Designing for pedestrians: a level of service concept", Ph.D. Thesis, Polytechnical Institute of Brooklyn, 1970, 15p.
- [15] Fruin, J. J. "Pedestrian Planning and Design". Metropolitan Association of Urban Designers and Environmental Planners, 1971, 206p.
- [16] Fu, L.; Song, W.; Lv, W.; Lo, S. "Simulation of exit selection behavior using least effort algorithm", *Transportation Research Procedia*, vol. 2, January 2014, pp. 533–540.
- [17] Galea, E. R. "A general approach to validating evacuation models with an application to EXODUS", *Journal of Fire Sciences*, vol. 16–6, November 1998, pp. 414–436.
- [18] Garrett, A.; Carnahan, B.; Muhdi, R.; Davis, J.; Dozier, G.; SanSoucie, M. P.; Hull, P. V.; Tinker, M. L. "Evacuation planning via evolutionary computation". In: Proceedings of the IEEE Congress on Evolutionary Computation, 2006, pp. 157–164.
- [19] Geraerts, R. "Planning short paths with clearance using explicit corridors". In: Proceedings of the IEEE International Conference on Robotics and Automation, 2010, pp. 1997–2004.
- [20] Geraerts, R.; Overmars, M. H. "The corridor map method: a general framework for real-time high-quality path planning". In: Proceedings of the IEEE International Conference on Robotics and Automation, 2007, pp. 1023–1028.
- [21] Gwynne, S.; Galea, E. R.; Owen, M.; Lawrence, P. J.; Filippidis, L. "A review of the methodologies used in the computer simulation of evacuation from the built environment", *Building and environment*, vol. 34–6, November 1999, pp. 741–749.
- [22] Hagan, M. T.; Demuth, H. B.; Beale, M. "Neural network design". PWS-Kent Publishing Company, 1996, 1012p.
- [23] Hansen, N. "A cma-es for mixed-integer nonlinear optimization", Ph.D. Thesis, French Institute for Research in Computer Science and Automation, 2011.
- [24] Hansen, N.; Ostermeier, A. "Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation". In: Proceedings of IEEE International Conference on Evolutionary Computation, 1996, pp. 312–317.
- [25] Helbing, D.; Keltsch, J.; Molnar, P. "Modelling the evolution of human trail systems", *Nature*, vol. 388–6637, July 1997, pp. 47–50.
- [26] Henderson, L. "The statistics of crowd fluids", *Nature*, vol. 229–5284, February 1971, pp. 381–383.

- [27] Henderson, L.; Lyons, D. "Sexual differences in human crowd motion", *Nature*, December 1972, pp. 353.
- [28] Henderson, L. F. "On the fluid mechanics of human crowd motion", *Transportation research*, vol. 8–6, December 1974, pp. 509–515.
- [29] Huang, P.; Kang, J.; Kider, J. T.; Sunshine-Hill, B.; McCaffrey, J. B.; Rios, D. V.; Badler, N. I. "Real-time evacuation simulation in mine interior model of smoke and action". In: Proceedings of the 23rd Annual Conference on Computer Animation and Social Agents, 2010, pp. 312–317.
- [30] International Maritime Organization. "Guidelines for evacuation analysis for new and existing passenger ships", Technical Report, 2007, 46p.
- [31] Ji, L.; Qian, Y.; Zeng, J.; Wang, M.; Xu, D.; Yan, Y.; Feng, S. "Simulation of evacuation characteristics using a 2-dimensional cellular automata model for pedestrian dynamics", *Journal of Applied Mathematics*, vol. 2013, August 2013, pp. 8.
- [32] Kennedy, J.; Eberhart, R. "Particle swarm optimization". In: Proceedings of IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
- [33] Liu, W.; Hu, K.; Yoon, S.; Pavlovic, V.; Faloutsos, P.; Kapadia, M. "Characterizing the relationship between environment layout and crowd movement using machine learning". In: Proceedings of the Tenth International Conference on Motion in Games, 2017, pp. 2.
- [34] Ma, W.; Yarlagadda, P. K. "Pedestrian dynamics in real and simulated world", *Journal of Urban Planning and Development*, vol. 141–3, July 2014, pp. 15.
- [35] Minsky, M. L.; Papert, S. A.; Bottou, L. "Perceptrons: An introduction to computational geometry". MIT press, 2017, 291p.
- [36] Nasir, F. M.; Sunar, M. S. "A survey on simulating real-time crowd simulation". In: Proceedings of the 4th International Conference on Interactive Digital Media,, 2015, pp. 1–5.
- [37] Polus, A.; Schofer, J. L.; Ushpiz, A. "Pedestrian flow and level of service", *Journal of transportation engineering*, vol. 109–1, January 1983, pp. 46–56.
- [38] Rosenblatt, F. "The perceptron: A probabilistic model for information storage and organization in the brain", *Psychological review*, vol. 65–6, November 1958, pp. 386.
- [39] Schadschneider, A.; Seyfried, A. "Empirical results for pedestrian dynamics and their implications for modeling", *Networks & heterogeneous media*, vol. 6–3, August 2011, pp. 545–560.
- [40] Snook, G. "Simplified 3d movement and pathfinding using navigation meshes". In: *Game Programming Gems*, Charles River Media, 2000, pp. 288–304.
- [41] Still, G. K. "Crowd dynamics", Ph.D. Thesis, University of Warwick, 2000.

- [42] Thalmann, D.; Musse, S. R. "Crowd Simulation". Springer-Verlag London Limited, 2007, 242p.
- [43] Tilley, A. R. "The measure of man and woman: human factors in design". John Wiley & Sons, 2002, 96p.
- [44] Treglia, D. "Game Programming Gems". Charles River Media, Inc., 2003, 614p.
- [45] Tsai, P.-C.; Chen, C.-M.; Chen, Y.-p. "PSO-based evacuation simulation framework". In: Proceedings of the IEEE Congress on Evolutionary Computation, 2014, pp. 1944–1950.
- [46] Van Den Berg, J.; Guy, S. J.; Lin, M.; Manocha, D. "Reciprocal n-body collision avoidance". In: Proceedings of the 14th International Symposium of Robotics research, 2011, pp. 3–19.
- [47] Van Den Berg, J.; Guy, S. J.; Lin, M.; Manocha, D. "Optimal reciprocal collision avoidance". Retrieved at <http://gamma.cs.unc.edu/ORCA/>, November 2016.
- [48] Van Toll, W.; Cook, A. F.; Geraerts, R. "Navigation meshes for realistic multi-layered environments". In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 3526–3532.
- [49] Van Toll, W.; Cook, A. F.; Geraerts, R. "A navigation mesh for dynamic environments", *Computer Animation and Virtual Worlds*, vol. 23–6, November 2012, pp. 535–546.



Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br