The Visual Computer (2019) 35:1119–1130
https://doi.org/10.1007/s00371-019-01684-9

ORIGINAL ARTICLE

CrowdEst: a method for estimating (and not simulating) crowd evacuation parameters in generic environments

Estêvão Testa[1] · Rodrigo C. Barros[1] · Soraia Raupp Musse[1]

Published online: 9 May 2019

Abstract

Evacuation plans have been historically used as a safety measure for the construction of buildings. The existing crowd simulators require fully modeled 3D environments and enough time to prepare and simulate scenarios, where the distribution and behavior of the crowd needs to be controlled. In addition, its population, routes or even doors and passages may change, so the 3D model and configurations have to be updated accordingly. This is a time-consuming task that commonly has to be addressed within the crowd simulators. With that in mind, we present a novel approach to estimate the resulting data of a given evacuation scenario without actually simulating it. For such, we divide the environment into smaller 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 develop a heuristic capable of aggregating per-room information so the full environment can be properly evaluated. Our method presents an average error of 5% when compared to evacuation time in a real-life environment. Our crowd estimator approach has several advantages, such as not requiring to model the 3D environment, nor learning how to use and configure a crowd simulator, which means any user can easily use it. Furthermore, the computational time to estimate evacuation data (inference time) is virtually zero, which is much better even when compared to the best-case scenario in a real-time crowd simulator.

Keywords Crowd simulation · Crowd estimation · Neural networks

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 predefined evacuation plans, but despite presenting strong similarities with real-world emergency scenarios [26], they still pose significant ethical, practical, and financial challenges to be addressed [12].

✉ Soraia Raupp Musse
soraia.musse@pucrs.br

Estêvão Testa
estevaostesta@gmail.com

Rodrigo C. Barros
rodrigo.barros@pucrs.br

1 School of Technology, Graduate Program in Computer Science, Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, Brazil

Crowd simulation is an interesting tool for evaluating real-world behavior of crowds in controlled scenarios. It can be defined as the process of simulating the movement of large amounts of agents, or crowds, in a previously defined environment. The different ways in which crowds can behave have been object of research for almost 30 years [25], including a variety of fields such as architecture, computer graphics, physics, robotics, safety engineering, training systems, psychology, and sociology [25,30].

Regarding evacuation planning, crowd simulation can be used to evaluate evacuation plans, given a parameterized environment. For instance, Cassol et al. [3] employed *CrowdSim* [2,10], which is a crowd simulator tested and validated in real-world scenarios, to create and gather data from simulations. They also employed CMA-ES [13,14], an evolutionary algorithm that varies the population data so that different portions of the crowd follow different routes.

Albeit simple, when considering all possible evacuation routes an environment can have, a very large amount of simulations have to be executed, growing exponentially as more details are added to that environment. Even though simulations can run in a similar time needed for real-world evacuation drills, it is still the most computationally demanding step in such an evacuation planning approach.

Hence, this paper presents a method whose goal is to estimate results of evacuation scenarios without the need of actually doing simulations in a complex environment within its run-time. For achieving that goal, we divide the environment to be analyzed into a set of smaller connected rooms. We hypothesize that by estimating the evacuation data on each room, we are capable of estimating the required information for the environment as a whole. For estimating per-room data, we make use of a machine learning approach, namely artificial neural networks (ANN), which are previously trained on data collected from a number of simulation scenarios simulated beforehand. Furthermore, we define a proper heuristic to unify per-room data in complex environments that comprise a set of connected rooms. To the best of our knowledge, this is the first work to make use of a hybrid machine learning and crowd simulation approach for the task of evacuation planning in generic environments and to compare results with real-life scenarios.

This paper is organized as follows. Section 2 presents related work on crowd simulation and evacuation planning. Section 3 describes our new approach to estimating parameters for evacuation plans based on machine learning. Section 4 reports the results of a detailed experimental analysis for validating our new approach, whereas in Sect. 5 we discuss our findings and point to future work directions.

2 Related work

The study and modeling of crowd traffic and their behavior, as well as the environment and evacuation characteristics is vital to the use of critical spaces [7,23,27,28]. We briefly review some existing papers in areas related to the three main aspects of this paper: crowd simulation, evacuation planning, and crowd learning. We are not exhaustive in our analysis since the majority of them are not focused in the exact same problem we address on this work, i.e., estimating crowd parameters instead of simulating it.

2.1 Crowd simulation

Several studies were proposed to elaborate ways of simulating crowds in egress scenarios.

SAFEgress (Social Agent For Egress) [5] 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 [20] is a specific application of agent modeling within the context of a virtual underground coal mine, with a fire and smoke propagation model, and a human physiology and behavioral model. The approaches described in [1,9,21] 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.

Van Den Berg et al. [34] propose the method *Optimal Reciprocal Collision Avoidance* ($ORCA$), which is used for robots to avoid collisions with each other. The method searches for the optimal velocity for each agent to move so all agents move through the environment without colliding. For that, it predicts the future positions of other agents and prioritizes velocities that minimize the probability of them colliding, ensuring that the agents adopt the velocities that will result in the lowest number of collisions by the time it performed the predictions.

Regarding evacuation planning, the work of Cassol et al. [3] makes use of crowd simulation (*CrowdSim*) and evolutionary strategy (CMA-ES) to search for the best configuration of routes for evacuation within a given environment. Similarly, Garrett et al. [11] employ evolutionary computation methods to evolve the placements of exits and other equipment in an effort to minimize the simulated evacuation time of the environment 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 Crowd learning

Tripathi et al. [32] recently surveyed techniques that have been used in the context of crowd analysis and convolutional neural networks. In particular, the crowd learning area is heavily focused on the crowd counting problem, also called density estimation. Chan et al. [4] create a privacy-preserving system for estimating the size of inhomogeneous crowds in a video. By segmenting the video and analyzing features detected between frames and regions, the number of people is estimated using Gaussian process regression. Similarly, Fradi et al. [6] extract features from videos and use a Gaussian symmetric kernel function to generate a crowd density map, allowing more specific locations of potentially crowded areas.

Liu et al. [22] is the closest approach to our work. 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. However, the authors simulate the crowd on a fixed environment (walls and routes) containing mobile parts whose disposition varies on 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.

In this work, we propose a methodology for estimating the evacuation of crowds in generic modular rooms using neural networks. These rooms can be connected to form larger and more complex environments, and we also propose heuristics to estimate this environment using the data obtained from estimating each particular room.

3 Proposed approach

In this section, we describe in details our proposed approach. First, we present the development of the crowd simulator that is employed to generate data for the learning process. Then, we present the process that was used to validate that simulator. Finally, we describe our per-room estimation strategy as well as the heuristics to unify the estimated results.

3.1 Crowd simulator

To develop the crowd simulator, we make use of the *Unity3D* game engine as a platform and implemented agents endowed with the main inherent collective behaviors as referred to in the literature: *(i) goal seeking*, *(ii) collision avoidance*, and *(iii) least-effort strategy* behaviors [8,16–19,29].

A *goal seeking* agent means that the agent will move inside the environment in order to reach a goal it is seeking. This behavior is simple to implement, just by keeping a goal position for each agent and moving the agent toward it.

When applying a *least-effort strategy*, the agent will move in trajectories that require less effort, avoiding turns while 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 native's from *Unity3D*.

When applying a *collision avoidance* behavior, the agent will move avoiding physical contact with obstacles and with other agents. In order to do that, we implemented the *ORCA* model [34]. The main goal of this method is to find the optimal velocity of each agent so that it manages to move through the environment without colliding with obstacles or agents. It corrects the course of their movements considering every other agent's velocity and position in a future time, reacting preemptively and avoiding possible collisions between them. We used as the optimization velocity for *ORCA* the direction each agent is following toward its goal on the *NavMesh*.

3.1.1 Model validation

For validating our crowd simulation, we compare our approach with the original ORCA method. For doing so, we recreated one of the *ORCA*'s showcases available at the developer's Web site [33]. Some images of this showcase can be seen in Fig. 1.

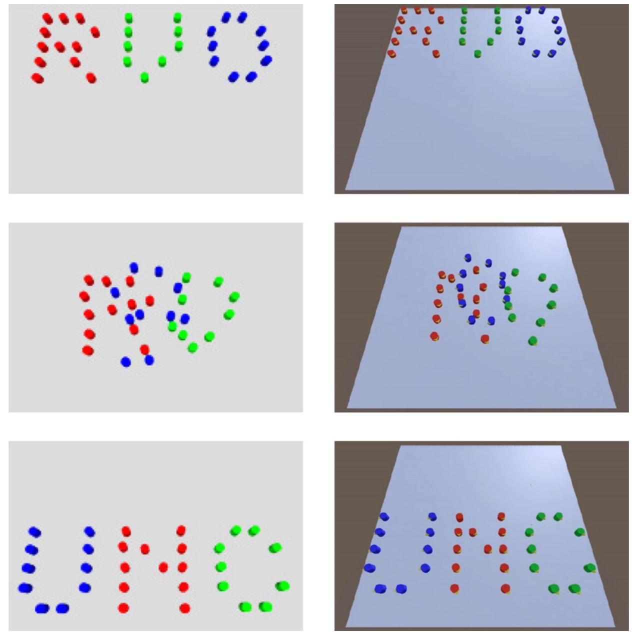


Fig. 1 Comparison between the *ORCA*'s showcase (left) and our simulation (right). A total of 41 agents are positioned to form the word "RVO", and then move down the screen to form the word "UNC" while avoiding collisions with each other

One of the showcases was created to depict the agents' behaviors using *ORCA* to avoid collision and its capacity to move efficiently while looking natural to the human eye. We recreated this showcase using solely visual information from real ORCA cases, and without knowing the agents parameters (e.g., agents positions or speeds), so we expect similar results from our model, though maybe not the same.

The showcase consists of 41 agents organized into three groups, where the agents from each group are positioned to form 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, whereas the one forming the letter "V" goes on to form the letter "C". Similarly, the one forming the letter "O" follows to form the letter "U". Easier movements could be made to complete this transformation; however, this specific movement was chosen in order to promote more potential collisions, thus requiring the use of collision avoidance mechanisms, which is 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 Fig. 1. In both cases, the agents succeeded in moving and forming the other word, in a quite similar way.

3.2 Estimation of rooms parameters based on ANNs

This section presents our model to estimate crowd data resulting from the evacuation process instead of having to simulate it. We are interested on testing artificial neural networks as a tool to estimate crowd parameters instead of generating them through simulations. The overview of our method comprises three phases: (i) performing several simulations in order to generate the training dataset for the ANN (see Sect. 3.2.1); (ii) ANN training and validation, as discussed in Sect. 3.2.2; (iii) testing the learning methodology for rooms estimation in complex environments (Sect. 3.3).

The room i parameters are described as follows:

- $Width_i$ is the room width (m);
- $Length_i$ is the room length (m);
- es_i is the exit size (m);
- f_i is the input flow: agents per second that enter within the room;
- F_i is the flow duration: duration in seconds defining the period where agents enter the room;
- ip_i is the initial population: number of agents which are inside the room at the beginning of the simulation.

Next, the estimation proceeds by estimating the *evacuation total time* (tt_i), in seconds, for the i^{th} room. Details on the estimation procedure is given in the following sections.

3.2.1 Dataset creation

To create a dataset of simulations that could be used to train and test the learning algorithms, we automatize 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 comprises a rectangular room with a single exit. In this paper, we use specifically the scope of rectangular rooms to represent the walkable space in a certain room and single exits in the training dataset. We decided to do so because if we were to include all possible geometries for rooms and all possible exit numbers, this would result in a combinatorial explosion of environment possibilities to be simulated.

Note that during inference time we can test non-rectangular rooms with more than one exit. For such, we have to approximate the room by a rectangle, and if we want to consider rooms with more than one exit, the user has to inform a single exit whose size is the sum of the size of all available doors. The room's dimensions vary according to the *room width* and *room length* parameters, whereas the exit size varies 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 the initial population are those already in the room at the beginning of the simulation. They are created and placed in a squared spiral pattern, forming a diamond at the center of the room as more agents are placed, to avoid agents overlapping

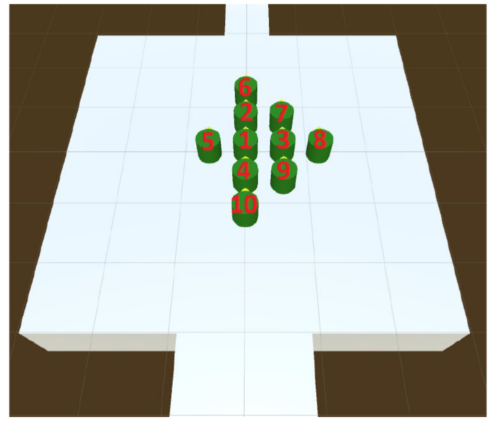


Fig. 2 Agents placed in a squared spiral pattern in a 6 × 6 room in the crowd simulator, numbered according to the order they are placed

each other. Figure 2 shows agents placed according to this pattern. As the number of agents rises, the area of the diamond increases. However, since their positions are restricted by the size of the room, if no space is available to accommodate all agents that are created it may be inevitable that some agents will start with overlapping positions. If this happens, agents will try to avoid collisions among them when the simulation starts, eventually finding enough space to move without colliding as other agents exit the room.

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

$$n_i = f_i \times F_i, \tag{1}$$

where f_i is the *input flow* defined for room i, and F_i is the *flow duration* for r_i. Indeed, we consider that the input flow in a certain room to be simulated is constant because we do not have more detailed information considering that we are only simulating a single room. If crowds are self-organized, which is what happens when people have space and time to adapt to the restricted physical space, their flow tend to be constant (e.g., already organized in a previous room in the environment hierarchy) [15,24].

Regarding the remaining parameters present the simulation: the space each agent occupies is represented through a cylinder with a radius of 0.3 m and it moves trying to maintain a max desired speed of 1.2 m/s. Both of those values are extracted from the literature [31].

To generate the training and validation sets, the parameters of the room were randomly defined within the intervals specified in Table 1. We generate 18,000 rooms for training and 2000 for validation. The tests are presented later in Sect. 4.

Table 1 Intervals for each parameter in the dataset generation process

Parameter	Min value	Max value	Value type
Room width ($width_i$)	2.0	20.0	Float
Room length ($length_i$)	2.0	20.0	Float
Exit size (es_i)	0.9	5.0	Float
Input flow (f_i)	1.0	10.0	Float
Flow duration (F_i)	0.2	100.0	Float
Initial population ip_i	0	99	Integer

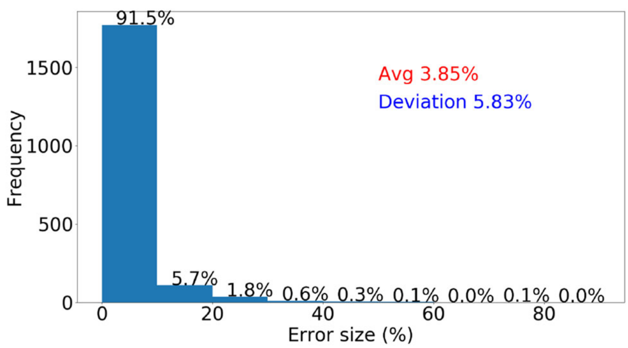


Fig. 3 Hidden layer with 400 neurons, total evacuation time error percentage

3.2.2 Training and validation

We train an ANN to estimate the evacuation total time tt_i. The training process was performed using the Stochastic Gradient Descent (SGD) optimization approach for 100 epochs with a $10e{-}7$ learning rate, stopping only when no more reduction on the loss was observed in new epochs even when reducing the learning rate. Once the tt_i is estimated, the validation errors were measured using the absolute relative error comparing the predicted value and the simulated value, here considered as the ground truth. We measured the quality of our estimations according to how many cases had less than 10% error.

We experiment with 1, 2, and 3 hidden layers with the 6 values of input from the dataset, 6 neurons on each layer and a single output value: the evacuation total time estimated. From those models, the one which achieved the best accuracy was the one with the single hidden layer. Then we proceeded to test several number of neurons on this layer. We tested with 2, 3, 6, 50, 200, 400 and 500 neurons on the hidden layer and we got the best validation accuracy with 400 neurons. Then, we proceeded to train the defined ANN stopping only when no more reduction on the loss was observed in new epochs, even when reducing the learning rate. The evacuation time in the 2000 rooms estimated by a 400 neurons ANN resulted in 91.5% of the validation cases below 10% error, as illustrated in Fig. 3.

The 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, we are now capable of estimating the time of evacuation for each room in the environment. However, crowd evacuation methods are used in general for more complex environments than a single room. Therefore, 3D modeling is usually necessary in order to create the environment to be simulated. Since we are proposing crowd estimation instead of simulation, we propose to create a simple environment based on a graph of rooms (i.e., an environment graph where the connections among the rooms are defined zas graph edges). We developed an environment editor (see Sect. 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 the last section) and then combine this data via empirically defined heuristics to compute the global data for the entire environment. Environment e comprises N rooms r to be estimated. We estimate the crowd parameters of each room r_i using the ANN for obtaining tt_i. Therefore, we have two types of rooms within an environment e: (i) rooms of type D, whose population impacts another room, i.e., people from r_{D_i} goes to another room; and (ii) rooms of type E (exit rooms), which are rooms that lead the population to the output of the environment. Moreover, those two types of rooms are mutually exclusive, i.e., rooms D never lead to exits and rooms E never impact any other room in e.

Figure 4 presents the overview of our estimation method. Note that some parameters are input for the rooms structure. For instance, r_k (room without dependent rooms, on the left) has the following input data: $width_k$, $length_k$, es_k (exit size) and ip_k (initial population). In addition, for this specific room the input flow $f_k = 0$ and flow duration $F_k = 0$ because no agents arrive in such space coming from other rooms. These are the input parameters for running the ANN and it results in tt_k (total time of simulation of r_k). In addition, variables $gfet_k$ (global first exit time) and fet_k (local first exit time) are computed using our heuristics (described later) and impact the next room in the hierarchy. r_i is a room with a dependent room (k) and it is also a dependent room regarding room j. Because r_i has a dependent room, f_i and F_i are no longer 0 and are impacted by r_k data, as shown in Fig. 4. It means that variables $gfet_k$ and fet_k are used to compute f_i and F_i, which are then used as input to the ANN together with the remaining input parameters: $Width_i$, $length_i$, es_i and ip_i to estimate tt_i. The same process happens for room j. On the right side of the figure, we illustrate the simulation pro-

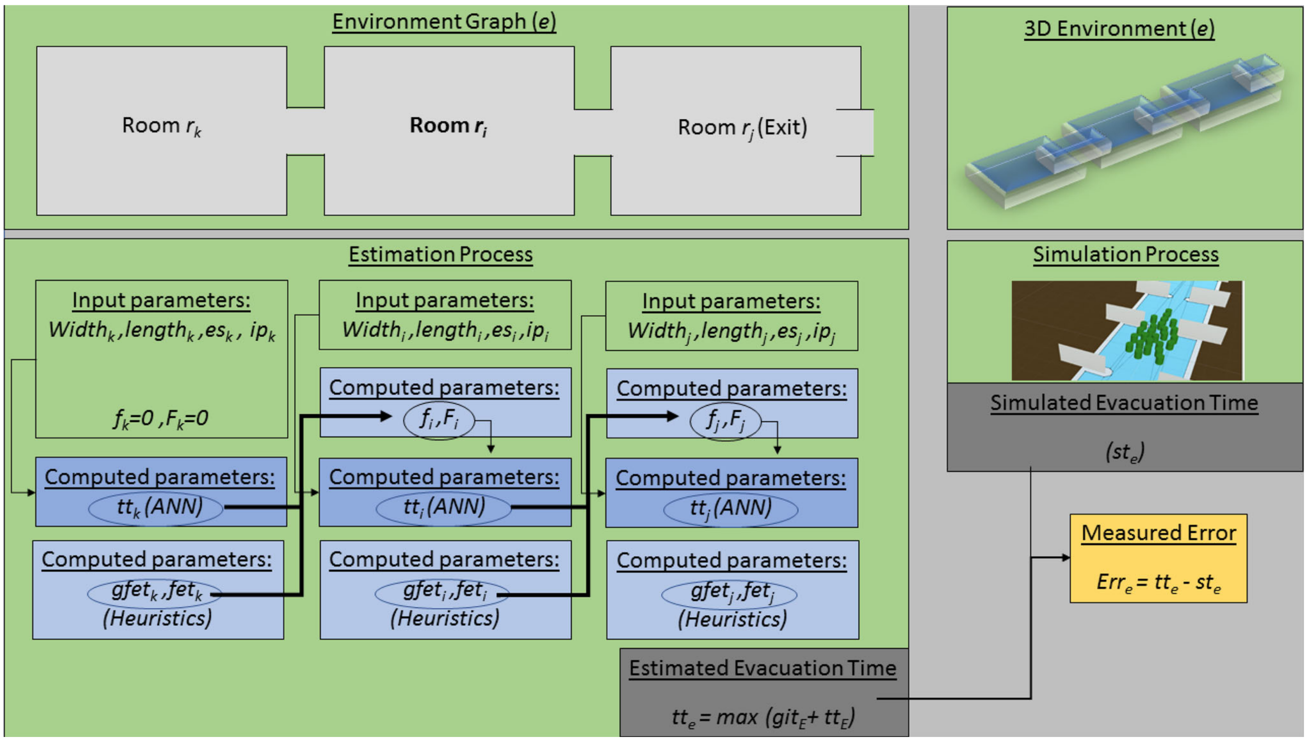


Fig. 4 Overview of the estimation method. The environment graph shows the hierarchy of rooms to be considered and how they affect other rooms. On the right, we have the same environment but modeled in 3D and simulated using our crowd simulation. In addition, the measured error is also illustrated

cess that happens using the 3D environment, and finally the measured error can evaluate the correctness of our estimation process (Err_e), where e is the simulated and estimated environment.

A Heuristics to Estimate per-Room Data

The environment estimator loads the specified environment (using the editor presented in Sect. 3.3.1) and performs the data estimation using the ANN for each room separately. The ANN outputs the estimated total time (tt) for each room. However, some rooms are dependent on others, i.e., a given room r_k (as illustrated in Fig. 4) is impacted by other rooms (type D) whose population moves toward r_k. This is the definition of the set r_{D_k} of dependence rooms that impact r_k. That is why some rooms have to be estimated before others. We deal with this problem by computing global and local-time parameters in order to synchronize the rooms. First we are going to present the execution flow for rooms that do not have dependence ($N_{D_k} = 0$), i.e., the first rooms within the environment graph.

1. Since nobody enters room k coming from another room (as illustrated in Fig. 4), then the *input flow* $f_k = 0$ and the *flow Duration* $F_k = 0$.
2. Once all parameters for r_k are set ($width_k, length_k, es_k, ip_k$ — those four coming from the Environment Editor — and $f_k = 0$ and $F_k = 0$), the estimation of r_k can happen normally. The neural network is loaded and the total time is estimated (tt_k).
3. In order to provide data regarding the agents that leave r_k and go to the next room, two other parameters are computed, fet_k and $gfet_k$, as follows:

$$fet_k = \begin{cases} 0, & ip_k = 0 \\ \dfrac{\frac{length_k}{2}}{MaxSpd}, & ip_k > 0, \end{cases} \tag{2}$$

where fet_i is the first exit time from r_k, meaning the time that the first agent exits that room, and ip_k is the number of agents initially created in room k. $fet_i = 0$ if there are no agents inside r_k. Conversely, if $ip_k > 0$, the first agent to exit will not need to walk throughout all the room length, so we consider half of the room length as an average for all agents to walk and exit the room. The next equation is computed afterward the ANN execution:

$$gfet_i = fet_i, \tag{3}$$

where $gfet_i$ is the time the first agent exited the room considering all rooms. For rooms without dependence, it is exactly the value of fet_i since they are those that "start" the crowd movement in the estimation. Therefore, fet_i

and $gfet_i$ are used to propagate values to the following rooms in the environment. In addition, pop is the final population considered in a given room, i.e., $pop = ip + (f.F)$, and for r_k without dependent rooms it is set as $pop_k = ip_k$.

For each room with $N_{D_i} > 0$ (see rooms i and j in Fig. 4), we perform the following execution flow: first, if the room has a dependence (r_{D_i}) that was not yet estimated, the estimation of this room is postponed until all dependents are estimated.

1. In order to estimate the population in r_i, we need to know the flow of people who enter in this room coming from other rooms. In our model, those concepts are represented through the following parameters: F_i (flow duration defined in Eq. 4) and f_i (input flow defined in Eq. 5), which consider data coming from dependent rooms (r_{D_i}) as follows:

$$F_i = (max(gfet_{D_i} - fet_{D_i} + tt_{D_i})) - min(gfet_{D_i}), \quad (4)$$

$$f_i = \frac{\sum_{i=1}^{N_{D_i}} (pop_{D_i}.in_{D_i})}{F_i}, \quad (5)$$

   where N_{D_i} is the number of dependent rooms of i, in_{D_i} is the percentage of agents from D_i that move to r_i, and pop_{D_i} is the population of r_{D_i}.
2. With the data from all dependents properly propagated for room i, the neural network can estimate the total evacuation time of such room (tt_i) as detailed before.
3. Finally, in order to propagate the agents parameters to the following rooms, parameters fet_i and $gfet_i$ are computed according to Eqs. 6 and 7 , respectively. Such equations are different from Eqs. 2 and 3 because room i has dependents, so pop is considered instead of ip.

$$fet_k = \begin{cases} 0, & pop_k = 0 \\ \dfrac{\frac{length_k}{2}}{MaxSpd}, & pop_k > 0, \end{cases} \quad (6)$$

$$gfet_i = fet_i + min(gfet_{D_i}). \quad (7)$$

After we finish the estimation of a certain room r_i, this entire process is repeated until all N rooms in the environment are estimated. Note that these simple equations have the unique goal of aggregating the flow of population passing through the rooms to compute the global time. The next section presents how such information is aggregated and used to estimate the entire environment.

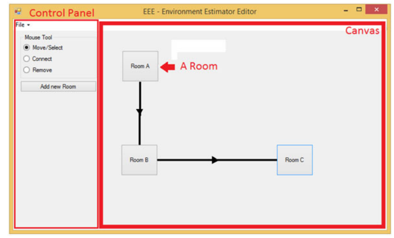


Fig. 5 Environment editor main window, with the control panel and the canvas regions. Arrows indicate the connections between rooms

B Aggregating per-room estimations

The last step of the environment estimation is to estimate the crowd parameter for the entire environment based on the per-rooms estimations. We 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 existing exit rooms that represent the exits in the environment, computed as:

$$tt_e = \max(git_{E_e} + tt_{E_e}), \quad (8)$$

where E_e are the exit rooms of the environment e, and tt_e is the obtained total time estimation using CrowdEst for environment e.

In order to evaluate the precision of our estimation, we compare tt_e with st_e, the latter being the simulation time obtained with our simulator as discussed in Sect. 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 the main window divided into two regions, the control panel and the canvas, as shown in Fig. 5. 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.

The editor allows for the following actions: (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 will move toward the second one; and (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 Fig. 6, 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 at any time.

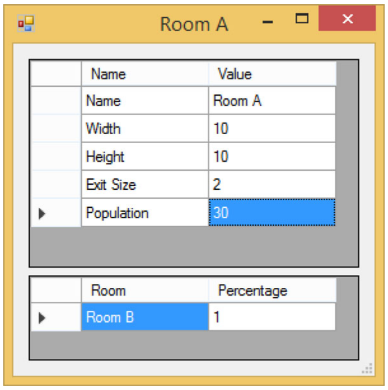


Fig. 6 Window for configuring room parameters within the editor. It shows the values of the parameters for a given elected room

4 Experimental results

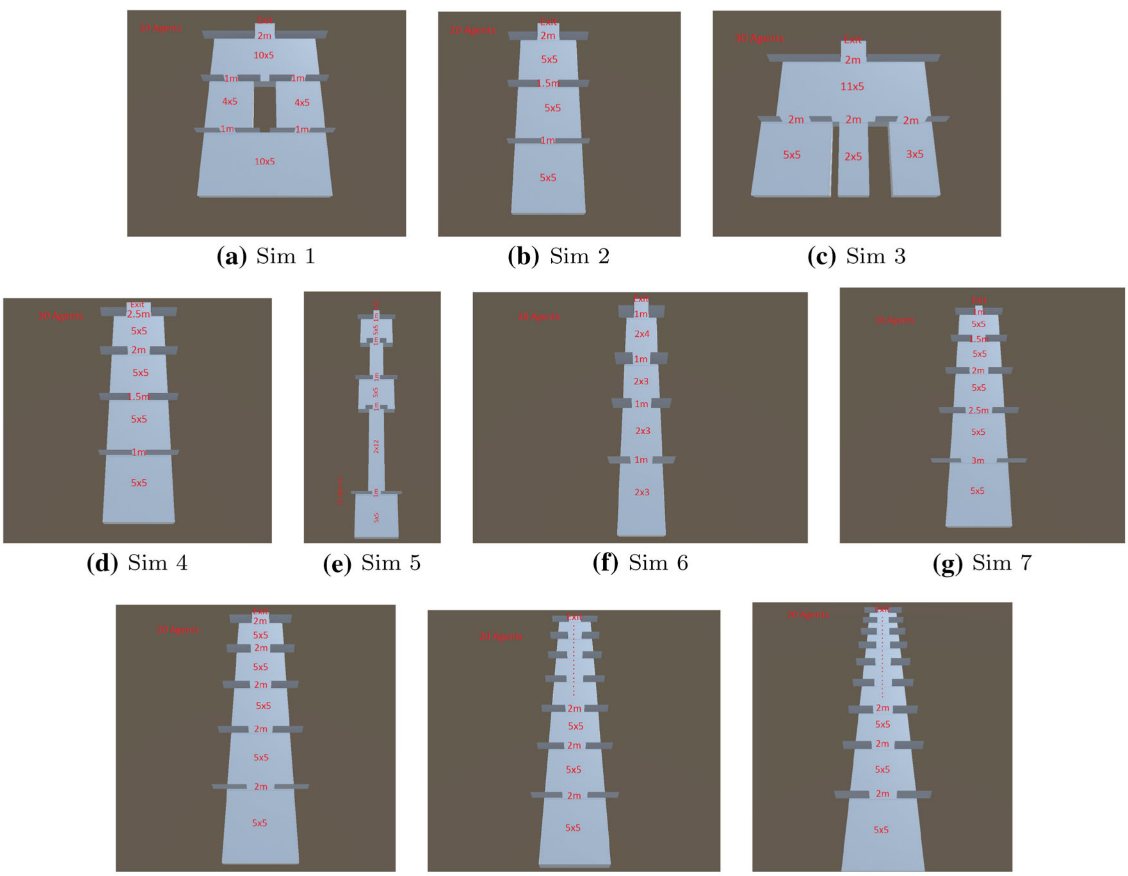
This section presents the experimental results. Section 4.1 describes the results when simulating and estimating 20 full scenarios with variable population distribution within the rooms. Then, we proceed with a practical usage of our method applying it in a real-life environment and comment about the usability of our application with real users.

4.1 Evaluating environments with varied populations

For the first set of scenarios to be tested, we create several distinct environments to be estimated and simulated. Our goal is to show the estimation errors when comparing our method for estimation with the full simulation execution.

First, using the Environment Editor, we have manually created 10 environments to be tested in our method using the application illustrated in Fig. 5. These environments, illustrated in Fig. 7, comprise a set of consecutive rooms, each of which with its own width, height, and exit sizes. In order to simulate those environments, we used our simulator developed at Unity. Figure 8 shows an example of an environment modeled using the Unity Engine (on the left) and using the Environment Editor (on the right).

For each of those environments, we configured two scenarios, one in which the initial population is placed only in



(a) Sim 1 **(b)** Sim 2 **(c)** Sim 3 **(d)** Sim 4 **(e)** Sim 5 **(f)** Sim 6 **(g)** Sim 7 **(h)** Sim 8 **(i)** Sim 9 **(j)** Sim 10

Fig. 7 Ten scenarios modeled in the crowd simulator for the 20 cases for evaluating environments. In the first 10 cases, the population is in the first rooms of the environments, while in the second set of simulation, the population is distributed along the rooms

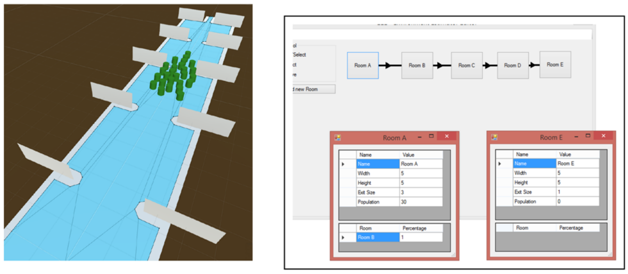


Fig. 8 Example of an environment case: five consecutive rooms with different exit sizes. On the left: the environment at Unity. On the right: the environment at the Environment Editor

the initial rooms of the environment (no dependents) and move toward the exit (last room), and other where the same number of agents is distributed throughout all rooms.

When analyzing the results, we note that the average errors for the total evacuation time tt for scenarios Sim1 to Sim10 is 20.9%, while the average error for scenarios Sim1' to Sim10' is 12.6%. The reason why the environments in which people are only placed in the initial rooms (Sim1 to Sim10) have the largest errors is because crowd simulators can take benefit from the self-organization of crowds. A population fixed in the initial rooms of the environment organizes themselves in lanes and other crowd structures from the beginning of the simulation, and does not need to change anymore in most of cases. CrowdEst, in turn, estimates each room separately without benefiting from the crowd previous organization, so we can say that the worst case when comparing crowd estimation and simulation is when the simulation takes benefit from the crowds' previous organization (Sim1 to Sim10).

4.2 Testing a practical example

In order to test our method in a practical example, we use the night club as described in [3]. We used the same 3D environment as illustrated in Fig. 9 to define the environment to be estimated using our editor and simulate it containing 240 agents, as described in [3].

The 3D environment has 3 floors and many rooms. There are rooms that have more than one exit (which is a feature that our method does not address), so we proceed by simply summing the widths of the doors to be the exit size value of the room and estimate such room using the ANN. Then, in the heuristic step, we propagate the percentage of population into their correct following rooms, since this information is present in the scenario. This step considers that the crowd is exiting by the doors at the same time and they finish at the same time. Such a decision makes it possible to simulate the night club (and any other environment) using our approach. Nevertheless, it can produce errors and should be better addressed in future work. Figure 10 shows the graph generated using the editor to model the night club (named SM).

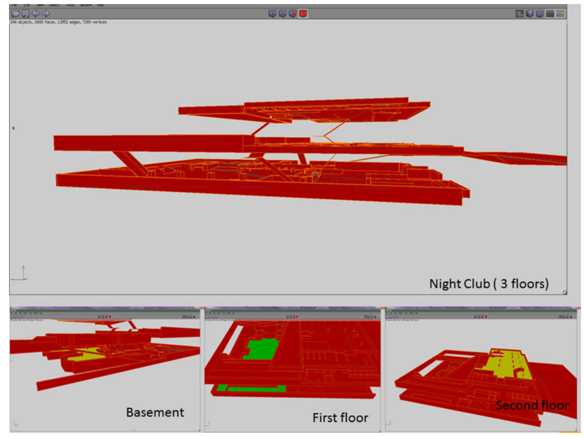


Fig. 9 Architecture of the nightclub used as base for simulation and estimation

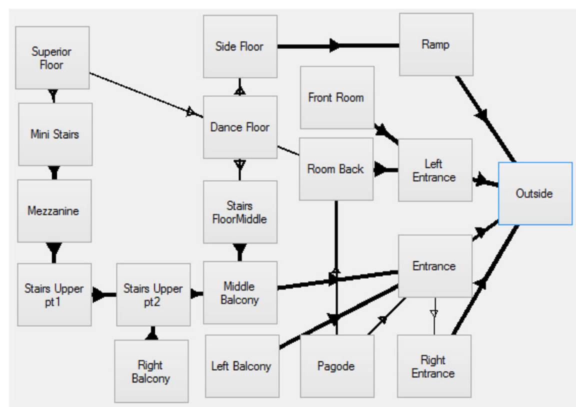


Fig. 10 Modeling of the SM night club in our editor

Table 2 shows the comparison of evacuation times tested in the night club with several methods. On the left, we have the evacuation time obtained in the real drill and simulated using CrowdSim (performed by Cassol et al. [3]). On the right we have results obtained with our work, CrowdEst. Note that our estimator presents an error of approximately +5% error in comparison with the real-life experiment. Hence, our simulation method achieves the lowest error when compared with our own estimation method and Cassol's simulator [3].

It is interesting to see that the previous work using CrowdSim achieves -18%, and it has been used with experts to guide the crowds in several real-life experiments, so the experts consider such an error to be acceptable in real-life scenarios. Each simulation/estimation have been executed 10 times and the average results are presented in Table 2.

In terms of advantages when using CrowdEst instead of simulations, we present the following remarks:

Table 2 Nightclub evacuation time obtained using different methods and comparison with the real-life drill

	Cassol et al. [3]		Our work	
	Real drill	Crowdsim	CrowdEst	OurSimul
Evac time	175 s	142 s	183 s	176 s
Relative error		− 18.85%	+4.57%	+0.57%
Compared with real drill				

1. Regarding the modeling of the environment:
   - 5–10 min was the time to model the night club in CrowdEst once the user has the floor plan with the detailed information needed from the environment;
   - 12–16 h is the time for a skillful designer to model the 3D environment, once s/he has the floor plan of the environment.
2. Regarding the environment and population configuration:
   - 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 [3]) to prepare the night club to be estimated can take weeks, and this needs to be repeated for each new environment to be simulated.
   - Using CrowdEst, we define the population and routes while we define the graph in the editor. Ten minutes was the time to accomplish this task and the task presented in item 1).
3. Regarding the computational time:
   - The environment estimation is virtually instantaneous, i.e., to estimate the ANN inference time takes less than 1 s (in CPU, since our method does not require a GPU).
   - As described in [3], a full simulation containing 240 agents in the night club executing at 24 frames/second takes around 3 min to evacuate the full environment.

As discussed in [3], we reinforce that if we want to simulate various distributions of people within the night club, our method can also be used. We can easily define the amount of people in the rooms, e.g., scripting it or using our editor, and instantaneously estimate the total evacuation time. Indeed, considering the night club that has three bifurcations, depending on the granularity used in the population distribution (e.g., 10% in exit and 90% in the other) we can have more than 1300 different plans to be executed. As mentioned in [3], approximately 80 from the 1300 possible plans with 240 agents took about 4 h, and 1010 agents took approximately 30 h to simulate. By using CrowdEst, we can easily and quickly find out the best evacuation plan considering only the evacuation time, among the 1331 plans, which takes approximately 22 mins of processing.

CrowdEst presents an error in comparison with real-life drills, but so does any crowd simulator. The main question is not really to achieve $error = 0$, considering that it is impossible even when comparing two different real populations evacuating the same real-life environment. The main intention is to obtain a realistic result not faraway from the expected one. CrowdEst obtains a 5% error in the only available practical example, which is a very promising result. In addition, if such results can be provided by a tool that can be easily tested, we can imagine one application where safety personnel can generate the environment graph, analyze the estimations, and make available the files for the population that is currently in the specific environment. Hence, people can learn and be properly trained for events in an easy and accurate way.

5 Final remarks

In this paper, we proposed a methodology for estimating the total evacuation time for complex and generic environments instead of simulating it. We used ANNs to train and estimate data for individual rooms, considering their geometry and population. Then, using such estimations, we propose heuristics to estimate the full environment parameters such as the total evacuation time. The approach we used to train ANNs over individual rooms and then aggregating data using simple heuristics seems quite promising, since we achieve an error of approximately 5% when comparing CrowdEst estimations with a real-life drill.

CrowdEst can save weeks (maybe months) of work when compared to the simulation approaches. We believe our tool is a further step in the direction to have all environments from real-life estimated and tested. Recall that this work is not about simulation, but estimation, so it explains why our paper does not have any attached videos, as commonly expected in crowd simulation research.

With respect to the limitations of the model: the training dataset was created solely with per-room information, containing only one entrance and one exit directly opposed to the entrance, which may not always be the case in the environments to be tested. We bypassed this problem increasing the width of doors. One way to improve the versatility of the model and to approach its usability for real cases is to add more parameters, which are important to define rooms such as multiple doors and exits. Obstacles are an entire issue by themselves, since there is a lot of information about them: shapes, sizes, positions, orientations, etc, for each obstacle present within the environment. In order to not lose too much of the abstraction of the model, we suggest creating a "level of obstruction" parameter for the rooms, with a value of 0 meaning the room is free of obstacles and can be traversed without problems, and increasing values for improving the obstacles difficulty on the movement of the agents. Defining this parameter would improve the similarity of the model with reality and we think it is worth a research path of its own.

As future work, we want to test CrowdEst in more real-life environments, besides modeling obstacles and increasing the number of input parameters for refining the ANN estimation.

Funding This work was partially funded by Brazilian Research Agency CNPq (Grant Number: 305084/2016-0).

Compliance with ethical standards

Conflict of interest The authors declared that they have no conflict of interest.

References

1. Aik, L.E., Choon, T.W.: Simulating evacuations with obstacles using a modified dynamic cellular automata mode. J. Appl. Math. (2012). https://doi.org/10.1155/2012/765270
2. Cassol, V., Oliveira, J., Musse, S.R., Badler, N.: Analyzing egress accuracy through the study of virtual and real crowds. In: 2016 IEEE Virtual Humans and Crowds for Immersive Environments (VHCIE), pp. 1–6 (2016). https://doi.org/10.1109/VHCIE.2016.7563565
3. 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 Comput. Graph. Appl. **37**(4), 60–71 (2017)
4. Chan, A.B., Liang, Z.S.J., Vasconcelos, N.: Privacy preserving crowd monitoring: counting people without people models or tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE, pp. 1–7 (2008)
5. Chu, M.L., Parigi, P., Law, K., Latombe, J.C.: Modeling social behaviors in an evacuation simulator. Comput. Anim. Virtual Worlds **25**(3–4), 373–382 (2014). https://doi.org/10.1002/cav.1595
6. Fradi, H., Dugelay, J.L.: Crowd density map estimation based on feature tracks. In: 2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP). IEEE, pp. 040–045 (2013)
7. Fruin, J.: Pedestrian Planning and Design. Metropolitan Association of Urban Designers and Environmental Planners (1971)
8. Fruin, J.J.: Designing for pedestrians: a level of service concept. Highw. Res. Rec. **355**, 1–15 (1971)
9. Fu, L., Song, W., Lv, W., Lo, S.: Simulation of exit selection behavior using least effort algorithm. Transp. Res. Proc. **2**(0), 533–540 (2014). https://doi.org/10.1016/j.trpro.2014.09.093. The Conference on Pedestrian and Evacuation Dynamics: (PED 2014), 22–24 October 2014. Delft, The Netherlands (2014)
10. Galea, E.R.: A general approach to validating evacuation models with an application to EXODUS. J. Fire Sci. **16**(6), 414–436 (1998)
11. 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: IEEE Congress on Evolutionary Computation, 2006. CEC 2006. IEEE, pp. 157–164 (2006)
12. 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. Build. Environ. **34**(6), 741–749 (1999)
13. Hansen, N.: A CMA-ES for Mixed-Integer Nonlinear Optimization. Tech. Rep. RR-7751, INRIA (2011)
14. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: IEEE International Conference on Evolutionary Computation, pp. 312–317 (1996)
15. Helbing, D., Buzna, L., Johansson, A., Werner, T.: Self-organized pedestrian crowd dynamics: experiments, simulations, and design solutions. Transp. Sci. **39**(1), 1–24 (2005). https://doi.org/10.1287/trsc.1040.0108
16. Helbing, D., Keltsch, J., Molnar, P.: Modelling the evolution of human trail systems. Nature **388**(6637), 47–50 (1997)
17. Henderson, L.: The statistics of crowd fluids. Nature **229**, 381–383 (1971)
18. Henderson, L., Lyons, D.: Sexual differences in human crowd motion. Nature **240**(5380), 353 (1972)
19. Henderson, L.F.: On the fluid mechanics of human crowd motion. Transp. Res. **8**(6), 509–515 (1974)
20. 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: The 23rd Annual Conference on Computer Animation and Social Agents (CASA), pp. 312–317 (2010)
21. 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. J. Appl. Math. (2013). https://doi.org/10.1155/2013/284721
22. Liu, W., Hu, K., Yoon, S., Pavlovic, V., Faloutsos, P., Kapadia, M.: Characterizing the relationship between environment layout and crowd movement using machine learning (2017)
23. Ma, W., Yarlagadda, P.K.: Pedestrian dynamics in real and simulated world. J. Urban Plan. Dev. **141**(3), 04014030 (2014)
24. Musse, S.R., Cassol, V.J., Jung, C.R.: Towards a quantitative approach for comparing crowds. Comput. Animat. Virtual Worlds **23**(1), 49–57 (2012). https://doi.org/10.1002/cav.1423
25. Nasir, F.M., Sunar, M.S.: A survey on simulating real-time crowd simulation. In: 2015 4th International Conference on Interactive Digital Media (ICIDM). IEEE, pp. 1–5 (2015)
26. Pauls, J.L., Jones, B.K.: Building evacuation: research methods and case studies. In: Canter, D. (ed.) Fires and Human Behaviour, pp. 227–249. Wiley, New York (1980)
27. Polus, A., Schofer, J.L., Ushpiz, A.: Pedestrian flow and level of service. J. Transp. Eng. **109**(1), 46–56 (1983)
28. Schadschneider, A., Seyfried, A.: Empirical results for pedestrian dynamics and their implications for modeling. Netw. Heterog. Med. **6**, 545–560 (2011)
29. Still, G.K.: Crowd dynamics. Ph.D. thesis, University of Warwick, Coventry, UK (2000)
30. Thalmann, D., Musse, S.R.: Crowd rendering. In: Crowd Simulation, pp. 195–227. Springer, London (2013)
31. Tilley, A.R.: The Measure of Man and Woman: Human Factors in Design, vol. 1. Wiley, New York (2002)

32. Tripathi, G., Singh, K., Vishwakarma, D.K.: Convolutional neural networks for crowd behaviour analysis: a survey. Vis. Comput. (2018). https://doi.org/10.1007/s00371-018-1499-5
33. Van Den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Optimal reciprocal collision avoidance. http://gamma.cs.unc.edu/ORCA/. Accessed 8 May 2019
34. van den Berg, J.P., Guy, S.J., Lin, M.C., Manocha, D.: Reciprocal n-body collision avoidance. In: Pradalier, C., Siegwart, R., Hirzinger, G. (eds.) Robotics Research. Springer Tracts in Advanced Robotics, vol. 70, pp. 3–19. Springer, Berlin, Heidelberg (2011)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Estêvão Testa is a Master in Computer Science at Pontifical Catholic University of Rio Grande do Sul (BR) where he was supervised by Prof. Soraia Musse. His research interests include crowd simulation, analysis, and estimation. Testa has also a B.S. in computer science from the Pontifical Catholic University of Rio Grande do Sul.

Rodrigo C. Barros received his Ph.D. on Computer Science from University of Sao Paulo in 2013. He received the Best Thesis in Computer Science Award from the Brazilian research agency CAPES, which is considered to be the most prestigious award for young scientists in Brazil. He also received the Best Thesis Award from the Brazilian Computer Society for his Ph.D. work. Dr. Barros is currently Associate Professor at the School of Technology, PUCRS, where he leads the Machine Intelligence and Robotics research group (MIR). He has published more than 90 peer-reviewed papers in journals and conferences, including in highly praised vehicles such as the Computer Vision and Pattern Recognition conference (CVPR), the International Conference on Machine Learning (ICML), the IEEE Transactions on Evolutionary Computation, just to mention a few. He is a co-founder of Teia Labs, the first AI-centered startup from the south of Brazil.

Soraia Raupp Musse is an associate professor at the School of Technology, Pontifical Catholic University of Rio Grande do Sul (Brazil), where she created and still coordinates the Virtual Human Laboratory. Her research interests include crowd simulation and analysis, facial animation, and integration of computer graphics, pattern recognition and computer vision. Dr. Musse has a Ph.D. in Computer Science from the Ecole Polytechnique Fèdérale de Lausanne (Switzerland) where she was supervised by Prof. Daniel Thalmann. Recently, she spent a year at UPENN (USA) working as a visiting scholar with Prof. Norman Badler. She has supervised more than 40 graduate students and postdocs and published more than 160 papers. She also published three books in Crowd Simulation with Springer-Verlag.