UCLouvain

epl

École polytechnique de Louvain

Design of an evacuation simulation tool for the "24h vélo" of Louvain-la-Neuve

Author: **Jeanne DELNESTE**
Supervisor: **Pierre SCHAUS**
Readers: **Hélène VERHAEGHE, Erwan MEUNIER**
Academic year 2024–2025
Master [120] in Computer Science and Engineering

Abstract

Managing large crowds during public events presents significant safety challenges, particularly in emergency situations. This study focuses on the development of a pedestrian simulation tool specifically designed for the *24h vélo of Louvain-la-Neuve*, one of Belgium's largest student events, which gathers over 40,000 people annually in a confined urban environment. The main objective is to support evacuation planning by identifying and evaluating pedestrian simulation models capable of realistically reproducing crowd behavior under high-density conditions.

To achieve this, four microscopic pedestrian models are mathematically described and compared using a consistent set of evaluation metrics, including realism, computational performance, and usability. An open-source platform, Vadere, was selected, modified, and used to implement the models and run the experiments. A two-phase experimental analysis was carried out: an initial comparison of all models, followed by a parameter optimization study on the most promising ones.

The results provide practical insights for event organizers and contribute to a methodological framework for evaluating pedestrian simulations. This work demonstrates the potential of simulation tools in enhancing crowd safety during large-scale events and offers a reproducible, adaptable foundation for further research and real-world applications.

Acknowledgments

I would like to sincerely thank my supervisor, Pierre Schaus, for his guidance and support throughout this year. His feedback and availability were a great help at every stage of this project.

I'm also deeply grateful to my family, who have supported and encouraged me not only during this year, but throughout my entire academic path. Their presence has always been a source of strength.

A big thank you to my friends (especially Eleonore) for being there during the long workdays, for the pep talks, and the laughs that helped me keep going. And finally, thank you to my boyfriend for his unwavering support, patience, and kindness during this intense period.

Table of Contents

Notations

$P_t(x)$	Target potential value at point x in the plane
g_i	Radius of the torso of pedestrian i
s_i	Intimate space width of pedestrians
s_p	Personal space width of pedestrians
$d_{p,l}(x)$	Euclidean distance between the center of pedestrian l and the position x
$P_{p,l}(x)$	Pedestrian potential value of pedestrian l affecting position x
R_o	Radius of influence of obstacles
$d_{o,j}(x)$	Euclidean distance between the closest point of obstacle j and position x
$P_{o,j}(x)$	Obstacle potential value of obstacle j affecting position x
$P_i(x)$	Potential value of position x for the pedestrian i
n	Number of pedestrians in the topography
m	Number of obstacles in the topography
$r(v)$	Step length of pedestrians according to their current speed
q	Number of equidistant points to consider on a circle in the discrete optimization
k	Number of circles to consider in the discrete optimization
v_i^{des}	Desired free flow speed of pedestrian i
$\vec{v}_i^{\text{des}}$	Desired free flow velocity of pedestrian i
λ	Time credit for the update schemes
$\vec{x}_i^{\text{des}}$	Desired destination for pedestrian i
$\vec{x}_i(t)$	Position of pedestrian i at time t
$\vec{e}_i(t)$	Desired direction of pedestrian i at time t
$\vec{v}_i(t)$	Actual velocity of pedestrian i at time t
τ_i	Relaxation time of pedestrian i
$\vec{F}_i^{\text{drv}}$	Driving force exerted on pedestrian i

$\vec{F}^{p}_{i,l}$	Pedestrian repulsive force exerted by pedestrian l on pedestrian i
$\vec{F}^{o}_{i,j}$	Obstacle repulsive force exerted by obstacle j on pedestrian i
$\vec{F}_i(t)$	Total motivation of pedestrian i
$\vec{N}_i$	Direction of motion of pedestrian i
$\vec{N}_{i,T}$	Direction of motion of pedestrian i influenced by its desired target
$\nabla P^{p}_{i,l}$	Gradient representing the influence of pedestrian l on pedestrian i
$\nabla P^{o}_{i,j}$	Gradient representing the influence of obstacle j on pedestrian i
$\vec{N}_{i,P}$	Direction of motion of pedestrian i influenced by other pedestrians and obstacles
R_p	Radius of influence of pedestrians
p_p	Scaling factor for pedestrians (GNM)
p_o	Scaling factor for obstacles (GNM)
ρ	Density of pedestrians
N	Number of pedestrians
b	Width of the measurement area
Δx	Length of the measurement area
J	Pedestrian flow
J_s	Specific pedestrian flow
A_i	Area of the Voronoi cell corresponding to pedestrian i
b_{cor}	Width of the bottleneck

Acronyms

AC	Average number of Collisions
CA	Cellular Automata
GNM	Gradient Navigation Model
GUI	Graphic User Interface
JSON	JavaScript Object Notation
MVC	Model-View-Controller
ODE	Ordinary Differential Equation
OSM	Optimal Step Model
SFM	Social Force Model
TC	Total number of Collisions
XML	Extensible Markup Language

Introduction 1

General Context and Motivation Managing large crowds is a serious challenge, especially during big public events. When thousands of people gather in one place, it becomes important to understand how they move, how they react to emergency situations, and how to keep them safe. This is why evacuation planning is a key part of event organization. In the last decade, researchers and safety professionals have been more and more interested in the study of crowd behavior ([Zha+11], [Rog+10]). With the help of computer models and simulations, it is now possible to better predict how people behave in dense environments, especially during emergencies.

The *24h vélo of Louvain-la-Neuve* is one of the biggest student events in Belgium. It attracts more than 40,000 people every year, including participants, visitors, and staff [Unind]. The event lasts 24 hours and takes place in the center of the city, which makes evacuation difficult in case of an emergency. The high density of people, the presence of alcohol, loud music, and night activities all add to the complexity of the situation. To improve safety, it is useful to simulate how people would behave in an evacuation. By testing different scenarios with a simulation tool, organizers can identify risks, improve their plans, and react faster if something goes wrong.

This work is motivated by the need to create such a simulation tool, specifically designed for the *24h vélo*. The goal is to understand how crowds move in this special context, and to find the simulation models that give the most realistic and useful results.

Fun fact: There are several categories for the bike race. One of the categories is the *folk bikes*, built by student groups. They are often designed to look like ships, tanks, or crazy machines, adding a creative and funny touch to the event.

Objectives The main objective of this work is to design and evaluate a simulation tool that can support the evacuation planning for big-scale events, such as the *24h vélo of Louvain-la-Neuve*. This tool should help organizers to better anticipate the movement of crowds in case of emergency and improve overall safety measures. To achieve this, the work is organized around several key goals:

- Studying and analyzing different pedestrian models.

- Defining a set of evaluation metrics that allows fair and meaningful comparison between these models, based on realism, performance and usability.
- Selecting and adapting an open-source simulation software to implement and test the chosen models.
- Performing a detailed experimental analysis to compare the models and determine which ones are best suited for the context of the *24h vélo*.
- Performing a parameter study of the most promising models to optimize their behavior for the specific conditions of the event.

By the end of this work, the aim is to provide a practical simulation tool that could contribute to safer crowd management of large-scale events.

Methodology This work follows a structured methodology including theoretical analysis, software adaptation, and empirical experimentations.

In Chapter 2, a mathematical description of four pedestrian models is carried out. This includes understanding how each model works, what assumptions it makes, and how it simulates pedestrian motion in a crowd. In Chapter 3, a set of metrics is defined to compare the models in a consistent way. These metrics include measures of realism, computational efficiency, stability, and how well the model handles high-density situations like those found during the *24h vélo*. Different open-source simulation frameworks are evaluated in Chapter 4, and one of them is selected. The software is analyzed in depth: its architecture, user interface, and model support. Several modifications are made to adapt it to the needs of this project. Once the software is ready, experiments are conducted in two phases in Chapter 5. First, all four models are compared using the defined metrics to identify their strengths and weaknesses. Based on this analysis, some models are excluded. Finally, the remaining models are studied in greater detail. Various parameters are adjusted to find the best configurations for specific characteristics of the *24h vélo* event.

This methodology ensures that the final result is based on both theoretical understanding and practical testing, making it a reliable support for real-world event planning.

My Contribution In this project, I contributed at several levels of the study. I first conducted a literature review on existing microscopic pedestrian models. Since the notations used in the different articles were not consistent, I created a unified mathematical description to make comparisons easier and clearer. I also searched for relevant metrics to evaluate the models in a fair and objective way.

To run the simulations, I explored various open-source tools and decided to use Vadere. I learned how to use its graphical user interface, studied its internal code structure, and modified some of its functionalities to better fit our needs. With the tool ready, I ran a comparative analysis of the four models based on the selected metrics.

After identifying the most promising models, I performed a parameter variation study to analyze how different settings influence their behavior. Through all these steps, I aimed to build a solid and practical comparison of pedestrian simulation models, with a focus on real-world applications such as crowd management during large public events.

All the experiments carried out are reproducible and the source code of the software with my modifications, simulations and data results are available at this address: `https://github.com/jedelneste/Memoire.git`.

Scope of the Work This work is useful both in theory and in practice. On the practical side, it can help event organizers improve crowd safety during large open-air gatherings taking place in different spots in a little city, such as the *24h vélo of Louvain-la-Neuve*. By testing and comparing different simulation models, this study can support better planning and faster response in case of an emergency evacuation.

From a scientific point of view, this project also gives a clear method for evaluating pedestrian simulation models. The selected metrics and the use of an open-source platform make the approach transparent and adaptable. Other researchers or engineers could reuse this work to test other models, or to apply it to different places or events.

Finally, this study opens the door to future improvements, like adding more realistic behavior to the models, or using real crowd data to calibrate the simulations.

Microscopic Crowd Simulation Models

2

Understanding and accurately modeling pedestrian crowd behavior is essential for planning and managing large public events. Ensuring the safety in case of an emergency is a top priority. In the last decade, researchers have developed many different mathematical models to simulate crowd movements in various environments ([SK12], [DK14], [HM95]).

Because there are numerous existing models, it is often difficult to decide which one to use in a specific situation. Each model has its own strengths and weaknesses, and not all models work well in every scenario. This makes the selection of the most suitable model an important step when planning real-life crowd management strategies.

All crowd simulation models can be classified into two main categories: **microscopic** and **macroscopic** models. While macroscopic models see the crowd as a continuum, microscopic models represent the crowd as a dense group of individual agents. Some recent researches combine elements from both methods to obtain hybrid models [KHB13], but this will not be discussed here.

Macroscopic Models They represent the crowd as a unified and continuous entity, and are mostly used to simulate very large-scale and dense scenarios. It is interesting when the goal is to observe the general moving pattern of the whole crowd. The basic principle behind these models is the analogy with fluid and particle motion, but the deep understanding of these models is out of the scope of this work.

Microscopic Models In microscopic crowd simulation models, each pedestrian is represented by an individual agent with its personal attributes. Each agent has its own position in space, its desired speed, its direction of movement and makes its own decisions. Their movement is influenced by nearby pedestrians and obstacles, and they avoid collision by reacting locally to their surrounding environment. The combination of these local behaviors produces individual final movement. These models are also called *Bottom-up* methods because they start by modeling small-scale interactions to create larger patterns.

Strengths and Weaknesses Macroscopic models are very useful and efficient when simulating large-scale and high-density scenarios. But the main drawback of macroscopic methods is the lack of realism and accuracy in dynamic situations. Indeed, as the individual features of people are not taken into account, the models do not capture the interactions of the individuals with the crowd. In some situations as in emergency situation, these interactions cannot be neglected.

The need of realistic results is the first and principal reason why we chose to investigate microscopic models in this work. Furthermore, microscopic methods are the most widely studied models to simulate crowds. These models capture the complexity of real-world phenomena such as bottlenecks, panic effect, lane formation, etc. due to the individual decision-making. Microscopic models are also more flexible. They can represent heterogeneous crowds with different profiles, for example children, elderly, or disabled individuals. Note that they are also more accurate for small-scale scenarios. On the other hand, microscopic models are usually more computationally expensive than macroscopic models. The challenge will be to find a microscopic model that can be efficient and realistic even with large crowds.

Chapter Overview In this chapter, we focus on four key models used in pedestrian dynamics: the Optimal Step Model (OSM), the Cellular Automata (CA), the Social Force Model (SFM) and the Gradient Navigation Model (GNM). Each model simulates pedestrian movement in its own way, with its own assumptions, parameters and mathematical formulations.

The Optimal Step Model focuses on how individuals choose their next step by considering possible positions and selecting the most favorable one based one some criteria. The Cellular Automaton model divides the space into a grid, where each cell follows simple rules to determine movement, making it less computationally expensive when simulating large crowds. The Social Force Model sees pedestrians as particles influenced by social forces, such as the desire to reach a destination or the need to maintain its personal space. Lastly, the Gradient Navigation Model uses gradients of distance functions to find the direction of the velocity vector which is then integrated to obtain the position, which allows smooth and realistic simulations without using force-based interactions.

In the following sections, we will describe mathematically each model, their foundations, their equations, and their parameters. We will also discuss how they represent pedestrian behavior. This mathematical analysis will provide a basis for comparing these models in later chapter, where we will assess their performance and discuss their parameters.

Common ground for all models All models have the same goal: simulating the movement of a group of agents (pedestrians) from a certain origin to a target, avoiding some obstacles and other agents. Agents can either be created and spawned from a source, or be placed on the plane before the simulation. There can be several targets and several sources. The four basic components; the agents, the origin(s), the target(s) and the obstacle(s), form what is called the *topography*. An example of scenario is given in Figure 2.1. We will always use the same colors for the topography: sources in green, targets in orange, obstacles in gray and agents in blue.

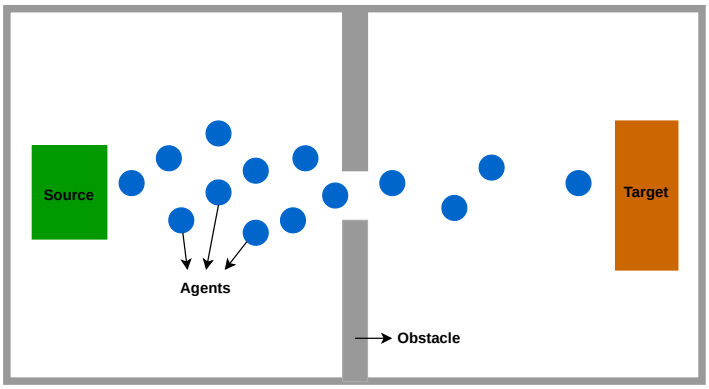


Figure 2.1: Example of topography

2.1 Model 1: the Optimal Step Model

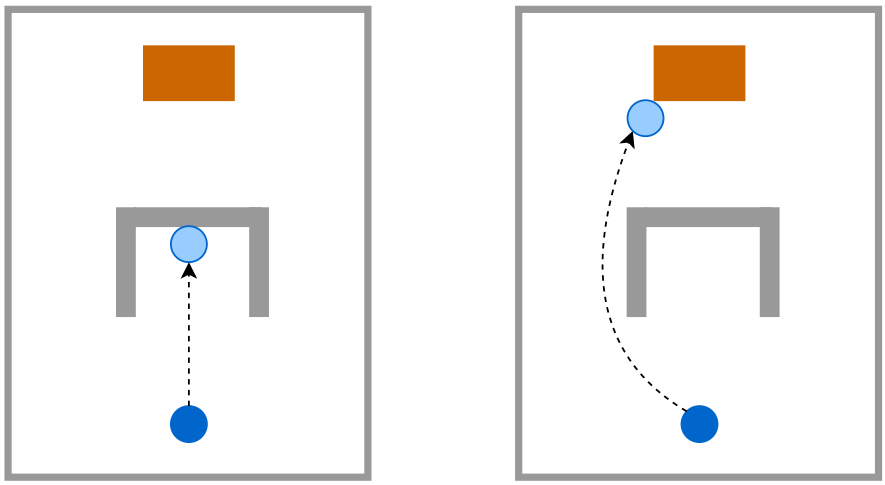
The basic idea of the **Optimal Step Model**, as described in the article [SK12], is that each agent tries to improve its situation locally at each time step according to some attractive and repulsive potentials. These potentials depend on the proximity to the targets, obstacles, or other agents. They are gathered into a scalar function called *utility*. The utility at each point will increase when approaching targets, and decrease when getting too close to obstacles or other agents. The principle is to compute the utility of each position in the topography, and each agent will choose its next step in a circle around it maximizing this utility.

Note that utility functions are defined as the negative potentials, because high values of potential are more repulsive while high values of utility represent attraction. This confusion comes from different notations in different articles.

2.1.1 Attractive and repulsive potentials

Different attractive and repulsive potentials are defined in order to create the utility function. This utility function will then allow pedestrians to find their best next step.

Target attractive potential To guide agents towards the targets, an attractive potential is used, also called *floor field* [Har10]. The floor field represents the propagation of a wave front in the topography emanating from the targets. The value of this floor field for each point in the plane corresponds to the distance between this point and the nearest target. To compute this value, it is not the Euclidean distance that is used, which is the shortest straight-line between the agent and the target, because agents could get trapped by concave obstacles as shown in Figure 2.2. The *geodesic distance*[1] is used instead, which is intuitively the shortest path between the agent and the desired target that avoids obstacles. The mathematical description of this distance is outside the scope of this work, but the reader can find out more about it in [Har10]. This approach is crucial to allow pedestrians to move around obstacles and not get stuck behind them.



(a) Euclidean distance **(b) Geodesic distance**

Figure 2.2: Problematic scenario

[1] https://www.sciencedirect.com/topics/computer-science/geodesic-distance

Note that the floor field is not computed for every point of the topography. Instead, the geodesic distance is computed for some points (i, j) in the plane, and a bilinear interpolation is done to find the **target potential value $P_t(x)$ for an arbitrary point $x \in \mathbb{R}^2$ in the topography**.

Pedestrian repulsive potential Two types of repulsive potentials are used in addition to the attractive potential to the targets: the repulsive potential of other agents, and the repulsive potential of obstacles. This allows to ensure that pedestrians keep a certain distance from one another, and that they do not walk too close to walls.

The potential induced by the presence of other pedestrians is based on a theory introduced by Hall [Gol69] that defines four distance zones around a person: the intimate zone (up to 45cm), the personal zone (from 45 to 120cm), the social zone (between 120 and 360cm) and the public zone (greater than 360cm). In this model the authors assumed that agents outside the personal zone of a pedestrian do not affect his/her path's choice, for computational efficiency reasons.

A high value of repulsive potential will be set for pedestrians standing in the intimate zone of other pedestrians (Equation 2.1 (2)), a little lower value will be set for pedestrians standing in the personal zone (Equation 2.1 (3)), and a zero potential value will be set for pedestrians outside the personal space (Equation 2.1 (4)). The torsos of pedestrians will also be taken into account, and if the distance between two pedestrians is lower than the sum of the two torsos, which would mean that the pedestrians step on each other, then the potential value is set to a even high value, the highest (Equation 2.1 (1)). This is to avoid pedestrians stepping on each other.

The torsos radii of pedestrians i and l are represented by g_i and g_l, the intimate and personal spaces width by s_i and s_p and the Euclidean distance between the center of agent l and the position x by $d_{p,l}(x)$. The **repulsive pedestrian potential** is obtained with the following equation [Pednd]:

$$P_{p,l}(x) = \begin{cases} \phi_1(d_{p,l}(x)) + \phi_2(d_{p,l}(x)) + \phi_3(d_{p,l}(x)) & \text{if } d_{p,l}(x) < g_i + g_l \quad (1) \\ \phi_1(d_{p,l}(x)) + \phi_2(d_{p,l}(x)) & \text{if } g_i + g_l \leq d_{p,l}(x) < s_i + g_i + g_l \quad (2) \\ \phi_1(d_{p,l}(x)) & \text{if } s_i + g_i + g_l \leq d_{p,l}(x) < s_p + g_i + g_l \quad (3) \\ 0 & \text{otherwise} \quad (4) \end{cases} \tag{2.1}$$

$P_{p,l}(x)$ represents then **the repulsive potential of pedestrian l affecting**

another pedestrian i at position x. The functions ϕ are defined below.

$$\phi_1(d_{p,l}(x)) = \mu \exp\left[4\left(\left(\frac{d_{p,l}(x)}{s_p + g_i + g_l}\right)^2 - 1\right)^{-1}\right] \tag{2.2}$$

$$\phi_2(d_{p,l}(x)) = \frac{\mu}{a_p} \exp\left[4\left(\left(\frac{d_{p,l}(x)}{s_i + g_i + g_l}\right)^2 - 1\right)^{-1}\right] \tag{2.3}$$

$$\phi_3(d_{p,l}(x)) = 10^3 \exp\left[\left(\left(\frac{d_{p,l}(x)}{g_i + g_l}\right)^2 - 1\right)^{-1}\right] \tag{2.4}$$

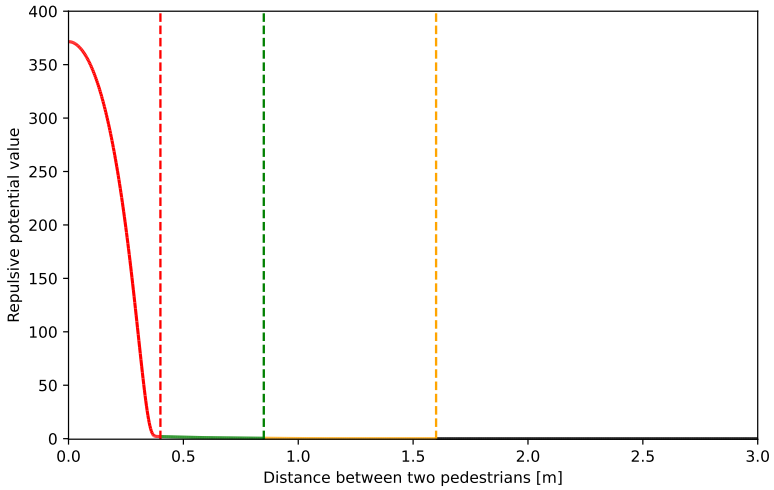
With μ and a_p some constants. Figure 2.3 shows the graphic representation of this pedestrian potential. One can clearly see the 4 parts of the equation on the graph. The red zone represents distances less than the torsos of the pedestrians, the green zone the intimate space and the orange zone the personal space. The second graph is a zoom on the intimate (green) and personal (orange) spaces, where one can see that the repulsion in the intimate space is higher than in the personal space.

Obstacle repulsive potential For the potential induced by obstacles, a distance R_o is defined, representing the scope of the obstacle repulsion. It is the same principle as for the pedestrian potential: if a pedestrian stands in the repulsion zone of the obstacle, the potential will be set to a specific value (Equation 2.5 (2)), and the potential will be zero for pedestrians outside the scope of repulsion (Equation 2.5 (3)). If two obstacles are near and there is a section common to both repulsion zones, the potential will be the sum of potentials from each obstacle in this section. The Euclidean distance to the closest point of the obstacle j from position x is defined as $d_{o,j}(x)$ and the radius of the torso of agent i as g_i. Again, the torso of pedestrians is taken into account and a very high value of potential is set if the distance between the pedestrian and the obstacle is lower than the pedestrian's torso (Equation 2.5 (1)).

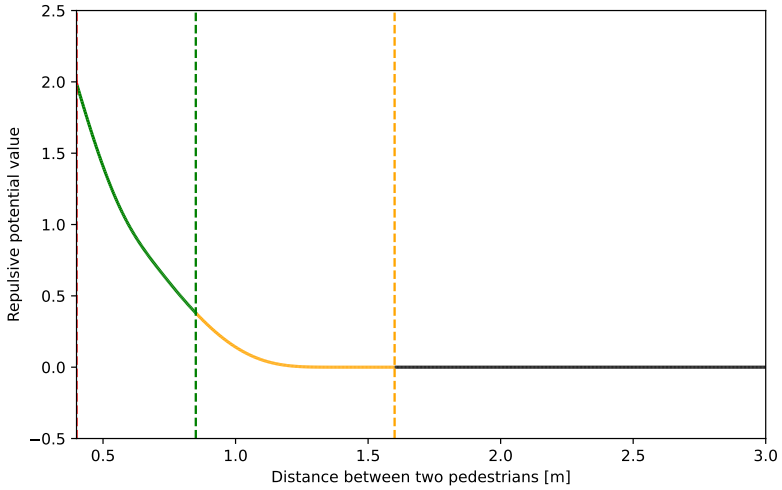
The **obstacle repulsive potential** is obtained with the following equation [Pednd]:

$$P_{o,j}(x) = \begin{cases} \delta_1(x) + \delta_2(x) & \text{if } d_{o,j}(x) \leq g_i \ (1) \\ \delta_1(x) & \text{if } g_i \leq d_{o,j}(x) \leq R_o \ (2) \\ 0 & \text{otherwise } (3) \end{cases} \tag{2.5}$$

$P_{o,j}(x)$ represents then the **repulsive potential of obstacle j affecting a**



(a) Repulsive pedestrian potential



(b) Repulsive pedestrian potential zoomed

Figure 2.3: Representation of the pedestrian repulsive potential

pedestrian i standing at position x. The functions δ are defined as:

$$\delta_1(x) = 6 \exp\left[2\left(\left(\frac{d_{o,j}(x)}{F_o}\right)^2 - 1\right)^{-1}\right] \tag{2.6}$$

$$\delta_2(x) = 10^5 \exp\left[2\left(\left(\frac{d_{o,j}(x)}{g_i}\right)^2 - 1\right)^{-1}\right] \tag{2.7}$$

The obstacle potential is rather weak to avoid some issues, for example small corridors. If the influence of obstacles were too strong, narrow passageways would become impassable because pedestrians would be too strongly repelled. To compensate the small intensity of the obstacle repulsion, the scope of the repelling R_o is larger. This allows pedestrians to navigate through small corridors while still being influenced by obstacles.

Total pedestrian potential All these potentials can be summed up in order to obtain, for each pedestrian i, a potential that can be evaluated at any point in the plane:

$$P_i(x) = \underbrace{P_t(x)}_{\text{target potential}} + \underbrace{\sum_{l=1, l\neq i}^{n} P_{p,l}(x)}_{\text{pedestrian potential}} + \underbrace{\sum_{j=1}^{m} P_{o,j}(x)}_{\text{obstacle potential}} \tag{2.8}$$

with n the number of pedestrians and m the number of obstacles in the topography.

2.1.2 Natural discretization on local circles

Now that the potential values for each point of the topography has been determined, the next problem to consider is how agents are going to choose their next step. The utility function to minimize is denoted as $-P$, because the higher the value of the potential function, the more repulsive the position is. The points on the target have an utility value of 0, the highest utility.

Since pedestrians move in discrete steps, the optimization is done locally for each step rather than planning a full path in advance. This method is called greedy, as it chooses the best next step without thinking about what might happen later. As the floor field provides global information about the environment, is assures that pedestrians wont get permanently stuck in local minima.

In order to find the best next step, each agent will look into positions within a circle around it as explained in the article [SK12]. A circular shape is used for simplicity, as shown in Figure 2.4, but other shapes (as ellipses, or individualized shapes) could easily be incorporated in the model. The radius $r \geq 0$ of this circle corresponds to the *step length* of each pedestrian.

An empirical experiment carried out in the same article [SK12] studied the relation between the step length and other parameters has shown that this step length is strongly related to the current speed v of the pedestrian. They came up with a linear regression model to compute the step length $r(v)$:

$$r(v) = \beta_0 + \beta_1 \cdot v + \epsilon \tag{2.9}$$

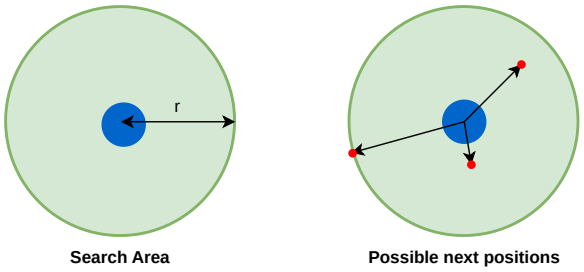


Figure 2.4: OSM: search area of agents

where β_0, β_1 are regression coefficients and the error term $\epsilon \sim N(0, \sigma)$ is normally distributed. The estimations of these coefficients found in the article are gathered in Table 2.1.

Parameter	Value
$\widehat{\beta_0}$	0.462
$\widehat{\beta_1}$	0.235
$\widehat{\sigma}$	0.036

Table 2.1: Default values for regression

As optimizing a non trivial function on a continuous circle is computationally expensive and require some sophisticated optimization algorithm, only a few discrete points on the circle are considered. A simple optimization approach involves selecting q equidistant points on a circle and taking the one with the lowest potential value, as shown in the left part of Figure 2.5. This method is computationally efficient when q is relatively small, typically between 8 and 32 according to the paper.

However, if the selected points always occupy the same fixed positions on the circle, it could introduce artifacts (unnatural movement patterns or biases) due to the rigid discretization of possible directions. To address this issue, the researchers added a small noise term:

$$\varphi = \frac{2\pi}{q}(z + u), \qquad u \sim U(0, 1) \tag{2.10}$$

with u a random value uniformly distributed between 0 and 1. Let $x_0 = (x_{1,0}, x_{2,0})$ be the current position of the pedestrian, the next position of the pedestrian will be the smallest value $P(x_z)$ for $z = 0, \ldots, q$ and

$$x_z = x_0 + r \times (\cos(\varphi), \sin(\varphi)) \tag{2.11}$$

with $x_z = (x_{1,z}, x_{2,z})$. Note that if the smallest value of potential corresponds to the current position x_0, the pedestrian will not move during this iteration.

This concept can be extended by also searching possible positions within the circle, in order to explore more possibilities for the next position. The optimization problem becomes two-dimensional. To solve it, a desired *number of circles* k is determined, and k circles are drawn inside the outer circle of radius $r \geq 0$ as seen in the right part of Figure 2.5. If r is the radius of the outermost circle, the radii of the circles are $r/k, 2 * r/k, \ldots, (k-1) * r/k, r$.

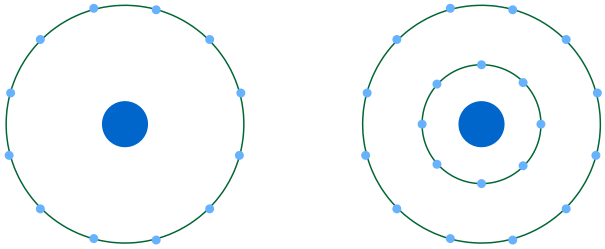


Figure 2.5: Discretization of the search area the simple way (left) and the extended way for k=2 (right).

Each pedestrian will then choose its next step among the evaluated points, according to the potential of each point. But a problem could occur: what happens when two pedestrians want to move to the same position at the same time? This will be solved by determining some update rules.

2.1.3 Update rules

One crucial issue is choosing the update scheme because it will schedule how agents move in the topography, and resolve conflicts such as if two pedestrians want to move at the same place at the same time. As time is divided into time steps, the positions of pedestrians can be updated by processing discrete events. The article [SK14] presents two ways to process these discrete events:

- **time-slicing approach**: moves forward in time by a certain time step Δt and then updates all elements in the topography.
- **discrete-event simulation approach**: processes the events at the simulation time they actually occur. It will be called the *event-driven* approach.

2.1.3.1 Time-slicing Approach

Two different updates are possible for the time-slicing approach: *fixed-order sequential* update or the *random shuffle* update. For the fixed-order update, the order of movement of pedestrians is determined by their creation time: the first created pedestrian will move first. It is implemented easily with a list data structure in which newly created pedestrians are added to the end of the list. It works well when pedestrians are created by a source, and thus not at the same time. This scheme has many advantages: it is easy to implement, there is no additional computation time due to some ordering of the pedestrians and collisions can be avoided. Furthermore, it represents a natural movement of pedestrians since the first created pedestrians will be closer to the target and less blocked by other pedestrians, so they will naturally move first. But this scheme also has its limitations, like for example when all pedestrians are created at once, or when several sources at different places create pedestrians at the same time.

The random shuffle update scheme is very similar, the only difference is that at each time step the order is randomly permuted. This can be seen as a sequential update with random order. The main advantage of this scheme is that the order of pedestrians in the list changes at each time step, reducing the biases caused by the order of movement. On the other hand, the drawback is that a random order is not meaningfully representative of real pedestrians' movements.

For both schemes, the update of pedestrian movement is done at each time step Δt which is given as a simulation parameter. All pedestrians are updated sequentially according to the order of the list (either fixed-order or random-order). In term of simulation time, all pedestrians are updated at the same moment, as seen in Figure 2.6.

Each pedestrian i has its personal *free-flow speed* v_i^{des} which corresponds to the desired speed when walking in a free plane, without any interaction. This desired speed is fixed, but can be adjusted in some cases, for example pedestrians slow down in dense crowds, or while walking through stairs. As the time is discretized, pedestrians must be informed at each time step of how much time has passed, and then they decide whether to move or not to maintain their desired speed.

This is done by introducing a time credit λ. The time credit increases at each update by the time step Δt, and decreases every time a move is made by the time taken to make this move (r/v_i^{des}). If a pedestrian does not have enough time credit to make a move (if $\lambda < r/v_i^{\text{des}}$), the move will not be allowed. This is how the different speeds of pedestrians are generated (see Algorithm 1).

Algorithm 1 Sequential Update Algorithm

$\lambda = \lambda + \Delta t$
if $\lambda \geq r/v_i^{\text{des}}$ **then**
    make move
    $\lambda = \lambda - r/v_i^{\text{des}}$
end if

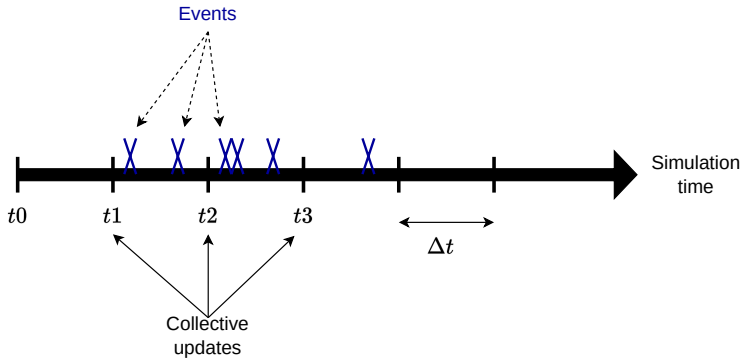


Figure 2.6: Illustration of the time-slicing approach

2.1.3.2 Event-driven Approach

In the event-driven update scheme, the events are processed in the natural order they occur: pedestrians are allowed to move as soon as they are done with their previous step, as illustrated in Figure 2.7. A priority queue is used to implement this scheme: the times of the next step of each pedestrian are computed, and sorted in increasing order in the queue. If two pedestrians want to reach the same place, the first in the queue will be allowed to move, and the second will adapt. Conflicts are solved in a natural way: first arrived, first served.

The only additional computation time of this scheme comparing to a fixed-order scheme is due to the insertion of events into the priority queue, which is limited.

2.1.3.3 Which one to choose?

As seen in [SK14], the chosen update scheme has an influence on the results of simulations. For the OSM, the event-driven update scheme is chosen as the reference scheme, because it represents natural behavior as the events are processed in the natural order they occur, and it is algorithmically simple.

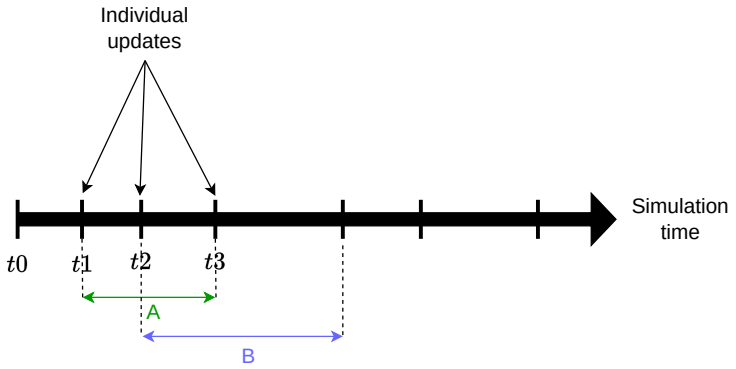


Figure 2.7: Illustration of the event-driven approach. In this scenario, the step for pedestrian B takes longer than for the pedestrian A.

2.2 Model 2: the Cellular Automata

[Wol83] describes **Cellular Automata** (CA) as *mathematical idealizations of physical systems in which space and time are discrete, and physical quantities take on a finite set of discrete values.* CA is a discrete time and discrete space simulation model. It is very similar to the OSM, by the use of attractive and repulsive potentials and a function called *utility*. The main differences will be the discretization of the topography and the chosen update scheme.

2.2.1 Structure of the topography

The topography area is divided into cells to create a grid, as seen in Figure 2.8. The cells must be regular for some practical issues, and must have the same size. Most implementations use rectangular grid, but there are other possibilities as triangular grid or hexagonal grid [Nit13].

Each cell of the grid can take a value from a finite set of possible states. For example, a cell can either be empty, or occupied with a pedestrian, an obstacle, a source or a target. At each time step, the state of each cell is updated taking into account only information from the neighboring cells. That is, agents move from cell to cell following some rules. The size of cells can be adjusted with a parameter, which is important as it will influence the discretization of the space and therefore the trajectories of pedestrians.

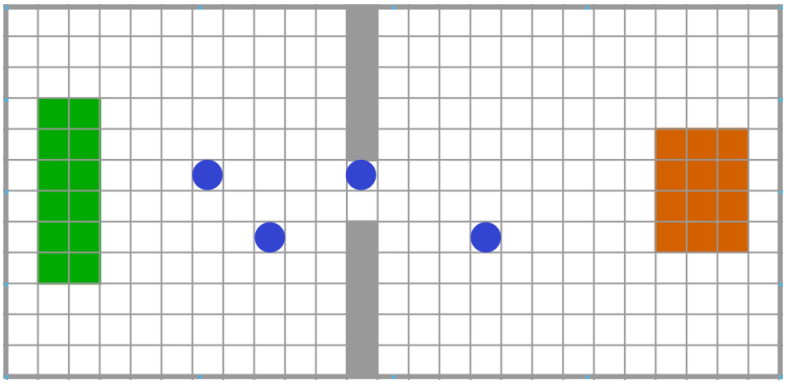


Figure 2.8: Grid structure of the topography

2.2.2 Pedestrian movement model

Different implementations of the CA algorithm exist, but in this work we decided to chose the one described in [Kle+19]. The same principle as for the Optimal Step Model is used to take the decision of where to make the next step. It was assumed that pedestrians can step forward in any direction using a fixed step length $r \geq 0$. A circle around the pedestrian is obtained, in which the pedestrian's next step will be chosen. The radius of this circle r is computed the same way as for OSM, with Equation 2.9.

As for the OSM, an utility value is assigned to each position in the topography using attractive and repulsive potentials. The same formulas are used for the attractive potential induced by the targets and the repulsive potentials from the obstacles and the other pedestrians.

Also, the same principle as OSM is used for the discretization of the step circle in order to find the point with the highest utility, see Section 2.1.2. The OSM can simulate the movement from grid to grid of the CA by choosing a specific combination of q and k, as shown in Figure 2.9.

The same issue as for the OSM has to be handled: what happens when two or more pedestrians try to reach the same cell at the same time? Some update rules are also discussed.

2.2.3 Update rules

[SK14] presented a new update rule for the CA model. As said in Section 2.1.3, the referenced method is the event-driven method. For CA, they proposed a parallel update scheme for event-driven update to try to improve the computational performance of the system, and then allowing to simulate large-scale scenarios.

Since discrete events are being handled, care must be taken when trying to parallelize the process. The goal is to ensure that the behavior of the model is not

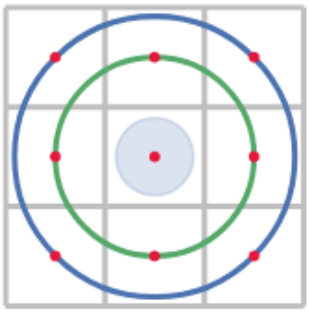


Figure 2.9: Illustration from [Kle+19] of the OSM mimicking a cellular automaton

altered. Only events that are independent from one another can be parallelized. But determining the non-conflicting events can be complex. For example, it might be assumed that two pedestrians standing far away from each other can be moved simultaneously, as they do not influence each other. But if there are pedestrians between them, they could indirectly affect each other through a chain of movement.

Another way to tackle the problem is to allow a parallel update of all pedestrians at the same time, but then the model would have to be adjusted. The model just has to be re-validated to be sure that this change only slightly impacts the results. The parallel scheme is composed of four important steps:

1. Pedestrians determine their next position without actually moving (**SEEK**)
2. Pedestrians move to their next position (**MOVE**)
3. Analyzing conflicts (**CONFLICTS**)
4. For each conflict, the pedestrian who was supposed to move first according to the event-driven update order remains at the new position while the other one is repositioned to his former position (**STEP**)

Note that the former positions of pedestrians that have to undo their step are always free because other pedestrians perceived this position as occupied while seeking their new position. Also, the time credit update is done after the last step, in order not to decrease the time credit of pedestrians that cannot move. The parallel process is describe in Algorithm 2.

2.3 Model 3: the Social Force Model

The **Social Force Model**, as presented in the article [HM95], is an Ordinary Differential Equation (ODE)-based model. The principle behind this model is that

Algorithm 2 Parallel Update Algorithm

```
λ = λ + Δt
if λ ≥ r/v^des then
    determine next position                          ▷ SEEK
    move to next position                            ▷ MOVE
    determine conflicts                              ▷ CONFLICTS
    undo step for conflicting pedestrians            ▷ STEP
    if pedestrian has moved to a new position then
        λ = λ − r/v^des
    end if
end if
```

every pedestrian is influenced by a set of *social forces*. Note that these forces are not actually exerted on the pedestrian's body, they are the representation of the motivation of the pedestrian to move. This motivation is then translated into an acceleration or a deceleration force reacting to the changes perceived by the pedestrian in his environment.

The Social Force Model defines three main forces that influence the pedestrian's movement: a driving force towards targets, a repulsive force from other pedestrians, and a repulsive force from obstacles.

2.3.1 The driving force

Each pedestrian i has a desired destination, denoted $\vec{x}_i^{\text{des}}$, which corresponds to the desired target. To reach their destination, it is assumed that pedestrians will take the shortest path possible $\vec{x}_i^1, \ldots, \vec{x}_i^n := \vec{x}_i^{\text{des}}$. If $\vec{x}_i^k$ is the next point to reach in the path, and $\vec{x}_i(t)$ denotes the actual position of pedestrian i at time t, the desired direction $\vec{e}_i(t)$ of a pedestrian is:

$$\vec{e}_i(t) := \frac{\vec{x}_i^k - \vec{x}_i(t)}{\|\vec{x}_i^k - \vec{x}_i(t)\|} \tag{2.12}$$

In certain cases, pedestrians aim to reach gates or area instead of points $\vec{x}_i^k$. The nearest point of the corresponding area is then taken at each time t, $\vec{x}_i^k(t)$. In a free plane, a pedestrian would walk towards its desired direction $\vec{e}_i(t)$ at a certain desired speed v_i^{des}. The desired velocity is then given by $\vec{v}_i^{\text{des}}(t) := v_i^{\text{des}}\vec{e}_i(t)$.

The actual velocity $\vec{v}_i(t)$ may not always match the desired velocity due to external factors like obstacles or other pedestrians. A relaxation time τ_i is introduced, which represents how quickly the pedestrian tries to return to the desired velocity.

The **driving force**, which is an acceleration term, takes the form:

$$\vec{F}_i^{\text{drv}}(\vec{v}_i, v_i^{\text{des}}\vec{e}_i) := \frac{1}{\tau_i}(v_i^{\text{des}}\vec{e}_i - \vec{v}_i) \tag{2.13}$$

A schema of this driving force can be seen in Figure 2.10.

Figure 2.10: Driving force

2.3.2 The pedestrian repulsive force

Pedestrians tend to stay at a minimal distance from each other, as someone would normally feel uncomfortable getting too close to an unknown person. This effect is translated by a repulsive force between pedestrians. The distance that pedestrians will try to keep between each other will depend on the density of pedestrians and their desired speed v_i^{des}. The private space of each pedestrian will also affect this distance. The repulsive effect of other pedestrians l is represented by a vector:

$$\vec{f}_{il}(\vec{d}_{i,l}) := -\nabla_{\vec{d}_{i,l}} V_{il}(b) \tag{2.14}$$

with $\vec{d}_{i,l} := \vec{x}_i - \vec{x}_l$ the distance vector between pedestrian i and l, and $V_{il}(b)$ a repulsive potential that is assumed to be a monotonically decreasing function of b. The equipotential lines (lines where the potential is constant) take the shape of ellipses oriented in the direction of motion. It is explained by the fact that pedestrians need lot of space in front of them when they are moving, and less space beside them. b represents the semi-minor axis of these ellipses, as shown in Figure 2.11, and can be obtained by:

$$2b := \sqrt{(\|\vec{d}_{i,l}\| + \|\vec{d}_{i,l} - v_l \Delta t \vec{e}_l\|)^2 - (v_l \Delta t)^2} \tag{2.15}$$

The quantity $s_l := v_l \Delta t$ is of the same order as the step width of pedestrian l.

$V_{il}(b)$ is known to be monotonically decreasing with b, so it can be expressed as an exponential function:

$$V_{il}(b) = V_{il}^0 \exp\left(-\frac{b}{\sigma}\right) \tag{2.16}$$

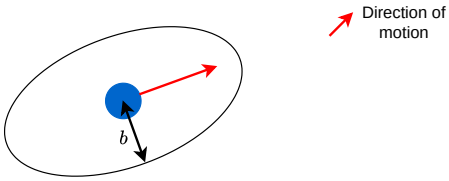


Figure 2.11: Illustration of the equipotential lines.

with some constant parameters V_{il}^0 and σ that can be determined experimentally. The values found in the article [HM95] will be set as default values, and are gathered in Table 2.2.

Parameter	Value	Unit
V_{il}^0	2.1	$m^2 s^{-2}$
σ	0.3	m

Table 2.2: Default values for the pedestrian potential

There is one issue left that has to be to tackled: the visibility of each pedestrian. Indeed, each pedestrian will prioritize what he sees in front of him. If an obstacle or another person is behind the pedestrian, its influence on the movement will be weaker. It is represented by introducing a scaling factor c with $0 < c < 1$.

This effect will depend on the desired direction $\vec{e}_i(t)$ of motion and the effective angle of sight 2φ, which means that agents inside this angle will affect the pedestrian normally but the effect of agents outside this angle will be reduced. The vectorial repulsive effect $\vec{f}_{il}$ will be adjusted with direction dependent weights:

$$w(\vec{e}, \vec{f}) := \begin{cases} 1 & \text{if } \vec{e} \cdot \vec{f} \geq \|\vec{f}\| \cos\varphi \\ c & \text{otherwise.} \end{cases} \tag{2.17}$$

The **pedestrian repulsive force** is obtained:

$$\vec{F}_{i,l}^{p}(\vec{e}_i, \vec{d}_{i,l}) := w(\vec{e}_i, -\vec{f}_{il})\vec{f}_{il}(\vec{d}_{i,l}) \tag{2.18}$$

2.3.3 The obstacle repulsive force

As for the previous repulsive force, pedestrians tend to keep a certain distance from obstacles as walls or borders of buildings. Similarly to the pedestrian repulsive force, a repulsive effect for each obstacle j of the topography can be defined. The **obstacle repulsive force** takes the form of:

$$\vec{F}_{i,j}^{o}(d_{i,j}) := -\nabla_{d_{i,j}} U_{ij}(\|\vec{d}_{i,j}\|) \tag{2.19}$$

with $U_{ij}(\|\vec{d_{i,j}}\|)$ a repulsive and monotonic decreasing potential. The distance vector between the pedestrian i and the closest point of the obstacle j is represented by the vector $\vec{d_{i,j}} := \vec{x}_i - \vec{x}_j^i$.

Again, the repulsive potential $U_{ij}(\|\vec{d_{i,j}}\|)$ ca be expressed as an exponential with some constant parameters:

$$U_{ij}(\|\vec{d_{i,j}}\|) = U_{ij}^0 \exp\left(-\frac{\|\vec{d_{i,j}}\|}{R}\right) \tag{2.20}$$

The default values of the parameters are found in the same article as for the pedestrian potential ([HM95]), and gathered in Table 2.3.

Parameter	**Value**	**Unit**
U_{ij}^0	10	$m^2 s^{-2}$
R	0.2	m

Table 2.3: Default values for the obstacle potential

2.3.4 Equation of motion for Social Force Model

As all effects described above influence a pedestrian's decision at the same moment, all the forces exerted on a pedestrian i at a time t can be summed up and **the total force $\vec{F}_i(t)$ influencing pedestrian i** can be found.

$$\vec{F}_i(t) := \underbrace{\vec{F}_i^{\text{drv}}(\vec{v}_i, v_i^{\text{des}}\vec{e}_i)}_{\text{driven force}} + \underbrace{\sum_{l=1, l\neq i}^{n} \vec{F}_{i,l}^{p}(\vec{e}_i, \vec{d_{i,l}})}_{\text{pedestrian repulsive force}} + \underbrace{\sum_{j=1}^{m} \vec{F}_{i,j}^{o}(\vec{e}_i, \vec{d_{i,j}})}_{\text{obstacle repulsive force}} \tag{2.21}$$

With n the number of pedestrians and m the number of obstacles. The first equation of the pedestrian dynamics model can be deduced from this equation. It represents the temporal changes of the preferred velocity for a pedestrian i:

$$\frac{d\vec{\omega}_i}{dt} := \vec{F}_i(t) + \text{fluctuations} \tag{2.22}$$

The *fluctuation term* represents random variations of the system. To complete this model, the equation of the temporal variation of the position $\vec{x}_i$ has to be derived. The preferred velocity $\vec{\omega}_i$ has been found with the previous equation, but

it has to be normalized because of a limitation of the actual pedestrian's speed, the *maximal acceptable speed* $v_i^{\max}$:

$$g\left(\frac{v_i^{\max}}{\|\vec{\omega}_i\|}\right) := \begin{cases} 1 & \text{if } \|\vec{\omega}_i\| \leq v_i^{\max} \\ v_i^{\max}/\|\vec{\omega}_i\| & \text{otherwise.} \end{cases} \tag{2.23}$$

Finally, the complete equations for the Social Force Model can be derived:

$$\frac{d\vec{\omega}_i}{dt} := \vec{F}_i(t) + \text{fluctuations} \tag{2.24}$$

$$\frac{d\vec{x}_i}{dt} = \vec{v}_i(t) := \vec{\omega}_i(t) g\left(\frac{v_i^{\max}}{\|\vec{\omega}_i\|}\right) \tag{2.25}$$

with $\vec{\omega}_i$ and $\vec{x}_i$ vectors representing the preferred velocity and the position of the pedestrian i. To compute each time the two components of these vectors, the SFM has to solve 4 EDOs.

2.4 Model 4: the Gradient Navigation Model

Instead of using physical forces to model movement, the basic idea behind the **Gradient Navigation Model** (GNM) is to use gradients to guide pedestrians toward their destinations, as explained in the article [DK14]. It is another ODE-based model, but differs from the Social Force Model by needing only 3 differential equations instead of 4. First the velocity vector is found by using a superposition of distance functions gradients, then this velocity is integrated to obtain the location. The concept of *floor field* is also used to create a navigation function, along with some local information. The principal idea is that pedestrians move toward the direction of steepest descent on this navigation function.

The goal of GNM is to determine the position $x_i \in \mathbb{R}^2$ of each pedestrian i at any point in time. A navigational vector $\vec{N}_i$ is introduced to describe a pedestrian's direction of motion. [DK14] uses 3 main assumptions to create the system of ordinary differential equations.

2.4.1 Assumptions of the model

Assumption 1 *"Crowd behavior in normal situations is governed by the decisions of the individuals rather than by physical, interpersonal contact forces."*

It is based on the observation that people generally avoid pushing each other in normal situations, they rather stand and wait. It allows to prioritize pedestrians decisions rather than the physical forces between them, like pushing or shoving. But this also makes the model less accurate in very high-density crowd situations.

Assumption 2 *"Pedestrians want to reach their respective targets in as little time as possible based on their information about the environment."*

This assumption is also used in other pedestrian models ([HM95], [KH17]). It states that each pedestrian will try to take the most straightforward path to its target.

Assumption 3 *"Pedestrians alter their desired velocity as a reaction to the presence of surrounding pedestrians and obstacles. They do so after a certain reaction time."*

In a free plane without any obstacles or other pedestrians, each pedestrian moves at its desired velocity. In real-life, they will adjust this velocity based on their surroundings.

2.4.2 Finding the Gradients

With the three assumptions stated above, [DK14] deduced that **the direction of motion of a pedestrian i, represented by the vector $\vec{N}_i$**, is influenced by three components: the pedestrian's desired target, obstacles and other pedestrians in the topography. The same concept of floor fields as for the OSM and CA is used. After finding this floor field, its gradient is computed. That will be the vector that points in the direction where the floor field decreases the fastest. Pedestrians will follow the opposite direction of this gradient, because they want to get closer to the goal and not farther away.

First thanks to **assumption 2**, the part of the **direction of motion $\vec{N}_{i,T}$ influenced by the target** can be computed. A function σ is defined, that represents by convention the opposite of the shortest distance from every point in the environment to the target, while avoiding obstacles. To build this field, the model solves an equation called Eikonal Equation[2]. The understanding of this equation is outside the scope of this work, but the reader can find out more about it in the article [TH16]. As said above, pedestrians will follow the opposite direction of the gradient of this function, because they want to get closer to their goal as fast as possible.

$$\vec{N}_{i,T} = -\nabla\sigma \tag{2.26}$$

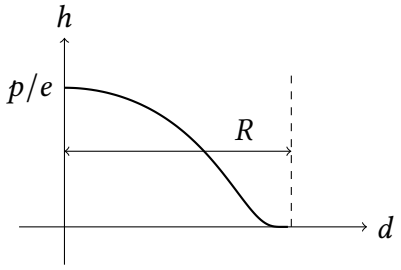
For the **assumption 3**, a vector $\vec{N}_{i,P}$ will be created, representing the **influence of other pedestrians and obstacles on the direction of motion of pedestrian i**. This vector will modify the direction of motion so that pedestrians keep a certain distance from obstacles and from each other.

[2] `https://www.sciencedirect.com/topics/earth-and-planetary-sciences/eikonal-equation`

To mathematically model this vector, the authors introduced a smooth function h that will represent the influence of one pedestrian l or one obstacle j on another pedestrian i, depending on the distance d between the pedestrian i and the obstacle j or other pedestrian l:

$$h(d; p, R) = \begin{cases} p \exp\left(\frac{1}{(d/R)^2 - 1}\right) & \text{if } |d/R| < 1 \\ 0 & \text{otherwise.} \end{cases} \tag{2.27}$$

R corresponds to the range of interaction, or support, which means that the influence is null beyond this radius, and p is a scaling factor. The larger p is, the stronger the interaction will be. When the distance d is equal to zero, the value of the function is the highest, giving approximately $p/e > 0$. The function is represented on the graph below.



This function will be slightly modified to avoid situations when pedestrians stand exactly on top of each other (i.e. the distance d is equal to zero), where the function h would have a too high value:

$$h_\epsilon(d; p, R) = h(d; p, R) - h(d; p, \epsilon) \tag{2.28}$$

with $\epsilon > 0$ a small constant. Note that $h_\epsilon(d; p, R) \longrightarrow h(d; p, R)$ for $\epsilon \longrightarrow 0$.

The viewing angle of pedestrians will also be taken into account. Indeed, pedestrians do not react equally to all their environment, they are more influenced by what they see. For that, the pedestrian influence will be scaled by a function $s_{i,l}$:

$$s_{i,l} = \tilde{g}(\cos(\kappa(\phi_{i,\sigma} - \phi_l))) \tag{2.29}$$

with $\tilde{g}$ a shifted logistic function[3], $(\phi_{i,\sigma} - \phi_l)$ the angle between the direction $\vec{N}_{i,T}$ which is the desired direction of pedestrian i and the vector from x_i to x_l. κ is a positive constant. The formulas for the gradients can now be derived. $\nabla P^p_{i,l}$ is the

[3] `https://en.wikipedia.org/wiki/Logistic_function`

gradient that represents the effect that pedestrian l has on pedestrian i and $\nabla P^o_{i,j}$ is the gradient that represents the effect of obstacle j on pedestrian i.

$$\nabla P^p_{i,l} = h_\epsilon(\|x_i - x_l\|; p_p, R_p) s_{i,l} \frac{x_l - x_i}{\|x_l - x_i\|} \tag{2.30}$$

$$\tag{2.31}$$

$$\nabla P^o_{i,j} = h_\epsilon(\|x_i - x_j\|; p_o, R_o) \frac{x_j - x_i}{\|x_j - x_i\|} \tag{2.32}$$

where x_i, x_l and x_j represent respectively the positions of pedestrian i, pedestrian l and obstacle j. R_p and R_o are respectively the radius of influence of other pedestrians and of obstacles, and p_p and p_o are positive scaling factors for the function h. These gradients have a norm that monotonically decreases with increasing distance. The equation can be decomposed into in three parts: the blue part correspond to the intensity of the repulsion, the red part, only for the pedestrian gradient, allows to taken into account the viewing angle of pedestrians, and the green part is responsible for the direction of the gradient. This direction points towards the other pedestrian or the obstacle. In order to avoid them, pedestrian will have to move in the opposite direction. That is why, to compute the final direction vector $\vec{N}_{i,P}$, the opposite of the gradients is taken:

$$\vec{N}_{i,P} = -\left(\underbrace{\sum_{l=1, l\neq i}^{n} \nabla P^p_{i,l}}_{\text{influence of pedestrians}} + \underbrace{\sum_{j=1}^{m} \nabla P^o_{i,j}}_{\text{influence of obstacles}} \right) \tag{2.33}$$

Now that the two vectors $\vec{N}_{i,T}$ (towards the target) and $\vec{N}_{i,P}$ (avoidance from people and obstacles) have been computed, they will be each scaled to have a length between 0 and 1. This helps to balance their influence on the pedestrian's decision. Without this scaling, one vector might be much stronger than the other just because of its size, not because it's more important. Even though the vectors are scaled, their direction is kept, so the pedestrian still knows where to go or what to avoid. After that, the sum of the two scaled vectors is scaled again. This second scaling will keep the combined result smooth, so the movement stays stable and realistic. With equations (2.26) and (2.33), the formula for **each pedestrian's desired direction** $\vec{N}_i$ is constructed:

$$\vec{N}_i = g(g(\vec{N}_{i,T}) + g(\vec{N}_{i,P})) \tag{2.34}$$

where $g : \mathbb{R}^2 \to \mathbb{R}^2$ is the scaling function, its exact equation can be found in the article [DK14].

2.4.3 Equations of Motion

The equations of motion for every pedestrian i can now be derived:

$$\dot{\vec{x}}_i(t) = \omega_i(t)\vec{N}_i(\vec{x}_i, t) \tag{2.35}$$

$$\tag{2.36}$$

$$\dot{\omega}_i = \frac{1}{\tau}\left(v_i^{\text{des}}\|\vec{N}_i(\vec{x}_i, t)\| - \omega_i(t)\right) \tag{2.37}$$

where $\vec{x}_i : \mathbb{R} \to \mathbb{R}^2$ represents the position of pedestrian i that will be found by integrating the velocity $\dot{\vec{x}}_i(t)$ with some numerical integration methods. $\omega_i : \mathbb{R} \to \mathbb{R}$ is the one-dimensional relaxed speed. Initial conditions $\vec{x}_0 = \vec{x}(0)$ and $\omega_0 = \omega(0)$, the initial positions and relaxed speeds, are added to the model. A reaction time τ that represents the time for pedestrians to adapt their actual speed to the relaxed speed ω is also introduced. Each pedestrian has its own desired speed v_i^{des} that generally depends on the local crowd density, but since the relation between velocity and density is still an open question, the authors simplified by choosing v_i^{des} constant and $v_i^{\text{des}} \sim N(1.34, 0.26)$. The values for mean and standard variations are based on several experiments, see [Wei93].

2.5 Comparative Analysis of Microscopic Models

We presented four of the most widely spread microscopic models: the Optimal Step Model, the Cellular Automata, the Social Force Model and the Gradient Navigation Model. The OSM and CA have many common points, but differ in their way to represent the topography. The other two, the SFM and GNM, are based on several differential equations to determine pedestrian's movement.

Discretization of the simulation Computers can only work step by step, so in order to run our simulation numerically, the time must be sliced into time steps. For the OSM and the CA, this discretization of the time is not a problem, because these models already work with time steps. One time step in the model will correspond to one time step in the simulation. The two other models will also have to be discretized. It will done by solving the ODEs at each time step Δt of the simulation. For the SFM, first the total force is computed, then the ODEs system is solved to find the preferred velocity at time $t + \Delta t$ based on the preferred velocity at time t. It ends the iteration by updating the positions. For the GNM, the iteration begins by computing the gradient and determining the direction of motion. Then the velocity is found by solving the ODEs and then the positions are also updated.

Cellular Automata Cellular Automata offers a computationally efficient method by simplifying pedestrian movement into discrete movements on a grid. This efficiency makes CA well-suited for large-scale, real-time simulations. However, the rigid spatial constraints of CA can lead to unrealistic pedestrian trajectories or flows, particularly in high-density scenarios where smooth, continuous movements are essential.

Social Force Model The Social Force Model is particularly useful for realistic simulations of dense crowds and emergency evacuations as it captures emergent pedestrian behaviors. However, it is highly computationally expensive due to the system of four ODEs to solve. While this model is highly effective for studying panic situations and social interactions, its complexity makes large-scale scenarios difficult to simulate.

Gradient Navigation Model The Gradient Navigation Model provides a more effective framework for large-scale situations, because it has one equation less to solve. However, its primary limitation lies in its lack of explicit modeling of pedestrian interactions, which can lead to less realistic behavior in highly dynamic or congested environments.

Optimal Step Model Finally, the Optimal Step Model offers a balance between computational efficiency and movement realism. By optimizing each step based on a cost function, it allows for fluid pedestrian motion while avoiding the complexities of ODE-based simulations. But because it prioritizes movement optimization rather than emergent behaviors, it may not fully capture self-organization phenomena seen in highly dynamic crowd situations.

Conclusion The selection of a crowd simulation model depends on the specific goals of the simulation. If the main objective is realism in pedestrian interactions, the Social Force Model provides the most accurate representation, but at a higher computational cost. For large-scale or real-time simulations where computational efficiency is the key, Cellular Automata serves as a practical choice, but it has its limitations in movement resolution. The Optimal Step Model provides a middle ground, allowing for smooth movement with reasonable computational efficiency, making it ideal for structured environments. Finally, the Gradient Navigation Model provides smooth trajectories at a reasonable computational cost but lacks detailed social interaction modeling. This is summarized in Table 2.4 with the legend:

✓ : Good

~ : Fair

✗ : Poor

	OSM	**CA**	**SFM**	**GNM**
Computational efficiency	✓	✓	✗	~
Smoothness of trajectories	~	✗	✓	✓
Simulating interactions	✓	✓	✓	~

Table 2.4: Summary of theoretical criteria

Evaluation Metrics | 3

Now that we found several different models to simulate the evacuation of pedestrians, one key question remains: which one is the most effective in our specific case? To answer this, we need a structured way of comparing them. Each model approaches pedestrian movement differently, making it essential to define a set of comparison metrics that objectively evaluates their performance, accuracy, and realism. These metrics will allow us to assess how well a model replicates real-world crowd behavior, how efficiently it simulates scenarios, and whether it meets the requirements of our application.

In our case, the concrete application is to simulate the evacuation of a large crowd of students attending a concert during the *24h vélo of Louvain-la-Neuve*. The places to evacuate are outdoor, enclosed by barriers and with multiple exits. These scenarios present specific challenges, such as congestion at bottlenecks (exits), crowd flow optimization, and behavioral responses under emergency.

This chapter explores some of the key metrics used to evaluate microscopic crowd simulation models, found in articles or deduced from real-life. We found a way to classify them into two categories:

- **Qualitative** or **visually validated** metrics: metrics requiring human interpretation or visual validation to assess the realism or accuracy of the model.
- **Quantitative** or **numerically defined** metrics: metrics based on precise numerical values and allowing for objective comparisons between models.

By establishing a clear framework for comparison, we can determine the strengths and limitations of each model and select the most appropriate one for our evacuation scenario.

3.1 Qualitative Metrics

Qualitative metrics are used to evaluate aspects of the simulation that require human interpretation or comparison with real-world behavior. These metrics often require visual analysis and human judgment to determine how realistic and natural the simulation results are. They do not always produce exact numerical values,

but they are important for understanding how closely the model reflects actual pedestrian behavior.

3.1.1 Trajectory Analysis

To ensure the realism of our models, we can look at the trajectories they simulate for the pedestrians. We could catch some unexpected behavior in these trajectories, such as pedestrians walking in straight lines or making detours before reaching their destination.

3.1.2 Behaviors in Specific Situations

Several crowd behaviors have been observed and studied through extensive and numerous experiments. Some of these crowd behaviors have been observed in very specific situations.

Lane Formation A common phenomenon is often observed in pedestrian dynamics: the emergence of organized pedestrians lane ([Zha+19]). This phenomenon is mostly seen in narrow streets or crosswalks, i.e. when pedestrians walk in both directions. But for our application we will simulate the evacuation of large places towards exits, so pedestrians will theoretically move all in the same direction. Therefore, this phenomenon will not be applicable in our case.

Bottleneck Dynamics An important behavior to study is the effect of narrow passages, or *bottlenecks*, on pedestrian movement. Studies have shown the influence of the bottleneck width on pedestrian flows and densities ([Lia+14], [KGS06], [Lid+11]). This will be important for our application because exits will be seen as bottlenecks and we want our simulations to be as realistic as possible. We can then observe the behavior of one exit with each of our models and discuss it.

3.1.3 The Fundamental Diagram

A well known but still open subject about crowd simulation modeling is the relation between the *speed* and the *density* of pedestrians. These two measures are gathered into a graph called the **fundamental diagram**, which represents the flow or velocity of pedestrians according to the pedestrian density. The average pedestrian density ρ over a rectangular measurement area $\Delta x \times b$ is defined as the number of pedestrians, ΔN, in the measurement area divided by its area:

$$\rho = \frac{\Delta N}{b \cdot \Delta x} \tag{3.1}$$

The flow J is defined in [Sey+09] as the number of pedestrians passing through the measurement area in a certain time interval, Δt.

$$J = \frac{\Delta N}{\Delta t} \tag{3.2}$$

In a corridor of width b, the flow can also be computed as:

$$J = \rho v b \tag{3.3}$$

with v the average speed of the pedestrian stream. If we replace the average speed v by $\Delta x/\Delta t$ and the average pedestrian density by the Equation 3.1, we find the Equation 3.2. The specific flow J_s is defined as the flow per unit-width and can be computed as follows:

$$J_s = \rho v \tag{3.4}$$

The specific flow allows to have results that do not depend on the size of the measurement area.

The fundamental diagram will differ for different situations like stairs, corridors or bottlenecks, and also if the crowd movement is uni- or bi-directional. We can cite the fundamental diagram of Weidmann [Wei93] which represents the simplest situation, a uni-directional movement in a plane without any bottleneck. Weidmann collected various data from various sources and came up with an equation for the speed-density relation:

$$v(\rho) = v_0 \left(1 - \exp\left(-g\left(\frac{1}{\rho} - \frac{1}{\rho_{\max}}\right)\right)\right) \tag{3.5}$$

where $v_0 = 1.34 m/s$ is the free walking speed, $\rho_{\max} = 5.4/m^2$ is the stand still density (largest density at which movement stops) and $g = 1.913/m^2$ is a constant called *gauge constant* by Weidmann. The specific flow J_s is then found with Equation 3.4. The fundamental diagram of Weidmann is shown in Figure 3.1.

This is the simplest and most used version of the fundamental diagram, but studies have shown that the flow-density relation can change according to a lot of different factors, from the age of participants to the experimental set-up, or again whether it is an emergency situation or not. It can even differ across cultures [CSC09]. An overview of fundamental diagrams can be found in [Kre19]. The only common point between all the existing fundamental diagrams is that the velocity tends to decrease with growing density.

A study has shown that the measurement methods can also influence the shape of the fundamental diagram [Zha+11]. This article presents 4 different methods to construct the fundamental diagram. The scenario used in the article is a uni-directional stream of pedestrians passing through a corridor.

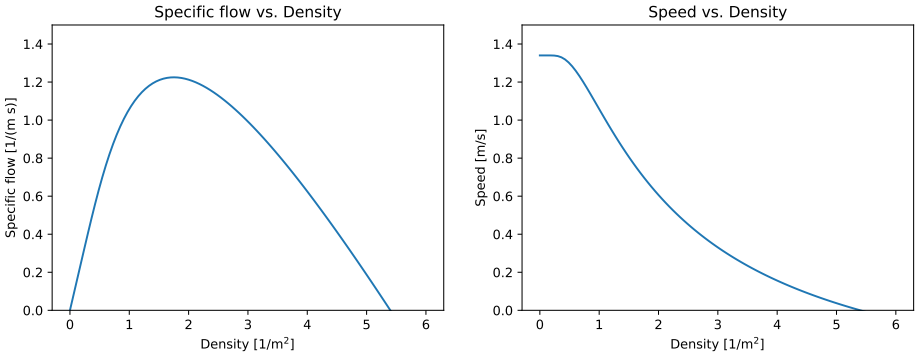


Figure 3.1: Fundamental diagram according to [Wei93]

Method A The first method looks at one specific spot in the corridor and counts how many people pass by over a certain amount of time. In other words, it takes a reference line x in the corridor and studies the mean value of flow and density over a fixed period of time Δt. It is represented in Figure 3.2. The number of

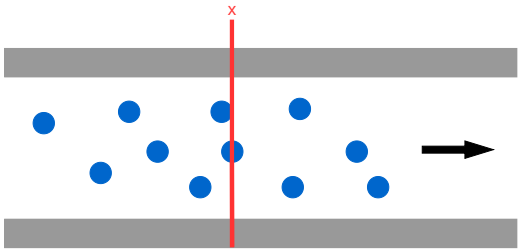


Figure 3.2: Method A

pedestrians $N_{\Delta t}$ passing the line x during the time interval Δt is computed, and $t_{N_{\Delta t}}$ is defined as the time between the first pedestrian crossing x and the last one. In other words, $t_{N_{\Delta t}}$ is the actual time took by the $N_{\Delta t}$ pedestrians to cross the line. With that the average pedestrian flow $\langle J \rangle_{\Delta t}$ is obtained:

$$\langle J \rangle_{\Delta t} = \frac{N_{\Delta t}}{t_{N_{\Delta t}}} \tag{3.6}$$

The average of the instantaneous velocities $v_i(t)$ of the $N_{\Delta t}$ pedestrians is computed to find the time mean velocity $\langle v \rangle_{\Delta t}$:

$$\langle v \rangle_{\Delta t} = \frac{1}{N_{\Delta t}} \sum_{i=1}^{N_{\Delta t}} v_i(t) \quad \text{with} \quad v_i(t) = \frac{x_i(t + \Delta t'/2) - x_i(t - \Delta t'/2)}{\Delta t'} \tag{3.7}$$

where $\Delta t'$ is a small time interval around t. The density ρ is then computed by:

$$\rho = \frac{\langle J \rangle_{\Delta t}}{\langle v \rangle_{\Delta t} \cdot b} \tag{3.8}$$

with b the width of the corridor.

Method B Intuitively, the second method watches how long each pedestrian takes to walk through a delimited section of the corridor. To compute the density, it checks how many pedestrians are in the section during that time. More formally, this method measures the average value of velocity and density for each pedestrian over space and time. Instead of taking a line as a reference, a measurement area of size $\Delta x \times b$ is taken as shown in Figure 3.3. The average velocity $\langle v \rangle_i$ of pedestrian

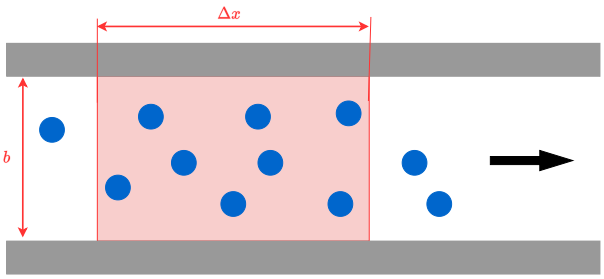


Figure 3.3: Method B

i is obtained by dividing the length of the measurement area Δx by the time the pedestrian takes to cross the area:

$$\langle v \rangle_i = \frac{\Delta x}{t_{\text{out}} - t_{\text{in}}} \tag{3.9}$$

The average density is computed for each pedestrian i as follows:

$$\langle \rho \rangle_i = \frac{1}{t_{\text{out}} - t_{\text{in}}} \cdot \int_{t_{\text{in}}}^{t_{\text{out}}} \frac{N'(t)}{b \cdot \Delta x} dt \tag{3.10}$$

with $N'(t)$ the number of pedestrians in the area at time t. The flow is found by taking the product of the density, the velocity and the width of the corridor:

$$J = \langle \rho \rangle_i \langle v \rangle_i b \tag{3.11}$$

Method C For the third method, it is as if a "picture" of the measurement area was taken, the same one as for Method B (Fig 3.3), at one moment in time. The data to observe will be how many pedestrians are in the measurement area and how fast each of them is moving right then. The average density $\langle \rho \rangle_{\Delta x}$ is defined as the number of pedestrians N divided by the area of the measurement section:

$$\langle \rho \rangle_{\Delta x} = \frac{N}{b \cdot \Delta x} \tag{3.12}$$

The spatial mean velocity is obtained by averaging the instantaneous velocities $v_i(t)$ of all pedestrians in the measurement area at time t:

$$\langle v \rangle_{\Delta x} = \frac{1}{N} \sum_{i=1}^{N} v_i(t) \tag{3.13}$$

Once again, the flow is computed with the product:

$$J = \langle \rho \rangle_{\Delta x} \langle v \rangle_{\Delta x} b \tag{3.14}$$

Method D The last method uses the Voronoi diagram, which is *a partition of a plane into regions close to each of a given set of objects.*[1] Here, the set of objects are the set of pedestrians, represented by a set of points, see Figure 3.4.

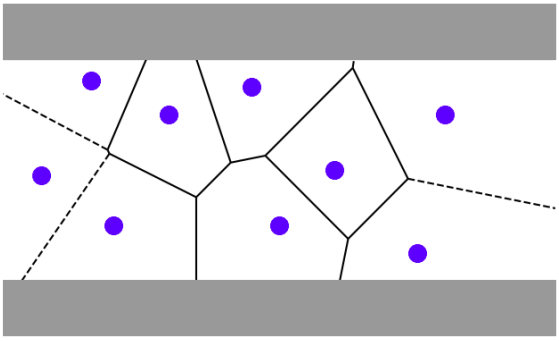


Figure 3.4: Method D

For each pedestrian i, the Voronoi cell area A_i is computed. The density ρ_{xy} and velocity v_{xy} distribution of the space can be defined as:

$$\rho_{xy} = \frac{1}{A_i} \quad \text{and} \quad v_{xy} = v_i(t) \quad \text{if } (x, y) \in A_i \tag{3.15}$$

[1] `https://en.wikipedia.org/wiki/Voronoi_diagram`

with $v_i(t)$ the instantaneous velocity of pedestrian i, see Equation 3.7. As for the Methods B and C, a measurement area of size $\Delta x \times b$ is defined and the average density and velocity are computed:

$$\langle \rho \rangle_v = \frac{\iint \rho_{xy} dx dy}{b \cdot \Delta x} \tag{3.16}$$

$$\langle v \rangle_v = \frac{\iint v_{xy} dx dy}{b \cdot \Delta x} \tag{3.17}$$

The flow is computed as follows:

$$J = \langle \rho \rangle_v \langle v \rangle_v b \tag{3.18}$$

In other words, this method gives each pedestrian its own space (like a bubble) based on how far they are from each other. It calculates their speed and how dense the crowd is in that little space.

Discussion All these methods compute the density, the velocity and the flow in order to build the fundamental diagram, but in different ways. We will investigate the influence of the measurement method on the diagram and discuss it.

3.1.4 Comparison with Real World

A great way to ensure the realism of our models is to compare them with real-world data like videos, of with well-controlled experiments. The drawback of this method is that it is difficult to find real-world data that perfectly matches our practical case.

Video Comparison The most accurate but difficult way to assess realism is to compare our data with real-world data, like videos of evacuation of stadiums or places. But these videos are rare, difficult to find and they generally do not match our practical case.

Experimental Validation Another good way is to compare our simulated data with well controlled experiments. The main issue is that it is difficult to reproduce emergency conditions in a laboratory, but we can still find good experiments. The comparison can be based on the *trajectory analysis*: we can measure the trajectory differences between models and observed paths. We can also try to reproduce the experimental set-up and compare the *evacuation time* of our models with the experiment. We can analyze the densities and speeds distributions along the topography as well.

3.2 Quantitative Metrics

Quantitative metrics rely on clearly defined numerical values. They provide objective and repeatable ways to compare the performance of simulation models. These metrics allow to identify strengths and weaknesses in terms of efficiency, accuracy and technical performance, such as collision avoidance, stability or scalability.

3.2.1 Pedestrian Reaching their Target

One simple but essential metric to evaluate is if all pedestrians always reach their desired target. For that, we can create different scenarios representing real-life situations, and test whether pedestrians manage to reach their destination avoiding the obstacles around them. In order to be the most complete possible, we have to imagine as much as different situations as possible, from the simple narrow street to the big open-air concert place. We can compute the proportion of the pedestrians that successfully arrive at their assigned target. Models that do not lead all pedestrians to their target will be easily rejected.

3.2.2 Stability of Evacuation Time

The evacuation time of a simulation, i.e. the total time for all agents to reach their specific target, is also a natural measure to look into. As the goal of the experimental part is to find the most realistic model and not the fastest one, we would need some experimental data to compare with to determine the best evacuation time, but we do not have them. We then want to know if the evacuation time is stable through different scenarios or if it radically changes with some perturbation in the initial conditions.

Indeed, to place a large number of pedestrians in the topography at the beginning of a simulation, we define a polygon in which a fixed number of pedestrians will be placed randomly. We want to know if this random placement of pedestrians influences or not the evacuation time. We will run several simulations of the same model with different random arrangements of the same amount of pedestrians in the same polygon and analyze the distribution of the evacuation time. This is important because when simulating the evacuation of large places, the emergency authorities won't be able to determine the exact real place of each real pedestrian, so they will have to place them at random, and the evacuation time is the most important measure when working in emergency situations.

3.2.3 Collision Avoidance

It is known in the literature ([DDH13]) that a natural human behavior is to avoid collisions while walking. Indeed, pedestrians would rather stand and wait behind

another pedestrian than run into him. This can be modeled by the *collision rate* which is defined as the number of agent-agent or agent-obstacle collisions per second. We can also measure the Total number of Collisions (TC) of the simulation and the Average number of Collisions (AC) per time step.

3.2.4 Scalability

As the goal of our models is to simulate places for the *24h vélo of Louvain-la-Neuve*, it is important that these model can handle a large number of pedestrians. Indeed, we are talking about thousands of students gathered in large places. We will investigate how efficiently our models simulate big crowds.

Execution Time The metric we will look into is the execution time for one simulation. As the different tested models are based on different mathematical principles and thus different implementations, their execution time will differ. We want to find a model that can simulate a scenario within a reasonable amount of time. We can also test the same model but with an increasing number of agents. As the models are microscopic and positions and velocities are computed at each time step for each agent, the execution time should increase with the number of pedestrians. We can also test what other parameters of each model influence the execution time as each model has specific simulation parameters.

3.3 Summary

There exists a lot of comparison metrics for microscopic crowd simulation models. To ensure objective and meaningful analysis, it is important to test different aspects of our simulations. Among all the metrics presented in this chapter, we won't be able to test them all, or we decided to set aside some metrics for different reasons. A summary of the metrics that we will keep and the metrics that will not be evaluated can be found in Table 5.4.

The chosen metrics allow us to assess how efficiently a model simulates evacuation dynamics, how realistically it replicates pedestrian behavior, and how feasible it is in terms of computational cost. By applying these metrics to our four described models, we will be able to identify which model provides the best balance between accuracy and efficiency for large-scale evacuations in constrained environments.

Metric	Keep?		Reason
	Yes	No	
Different Scenario	×		
Trajectory Analysis	×		
Lane Formation		×	Not applicable to our practical case
Bottleneck Dynamics	×		
Fundamental Diagram	×		
Video Comparison		×	Too difficult to collect
Experimental Validation	×		
Stability of Evacuation Time	×		
Collision Avoidance	×		
Scalability	×		

Table 3.1: Choice of Metrics

Vadere: an open-source simulation framework

4

After finding the models we will use to simulate the movements of our pedestrians, we had to find a software to implement these models, to test them and to compare them. We looked for an open-source framework in order to be able to modify it at our will. The reader will find in this chapter the description of the chosen framework and why we chose it, a description of its architecture, how models are implemented in it, and a description of its Graphic User Interface (GUI), which is the part of the software with which users will interact. In the last section, Section 4.5, we explain all the modifications added to the software in order to make its use easier.

4.1 Overview of Open-source Simulation Frameworks

When choosing a simulation software for running pedestrians' simulations, several important criteria must be considered. First of all, the software must be open-source to allow us to add functionalities or modify some things at our will. It should support microscopic models, as our four simulation models are all microscopic. It should also offer the implementation of several models to allow us to test and compare our four models in the same framework. Moreover, easy integration of new models is a key point, so that we can add without difficulty any model that is not yet implemented. Furthermore, the chosen software should allow the comparison of different models under similar conditions, to objectively evaluate their performance and accuracy. The presence of measurement tools to analyze results directly within the software is a plus. Finally, a GUI that is simple and intuitive would also be an additional advantage, as our first goal is to create an intuitive tool for the emergency authorities.

We found several existing software. We focused on three of them: Menge [CBM16], JuPedSim [Wag+15], and Vadere [Kle+19]. They are all open-source frameworks created for microscopic simulations of pedestrians and crowd movements. They are designed with a modular architecture, which makes it easy to extend or adapt the frameworks for specific needs. Each software already includes several pedestrian models and offers some form of visualization and data analysis to help users study pedestrian behavior. All three are built for an academic use, allowing

to freely modify and expand the source code.

4.1.1 Menge

Menge[1], developed at the University of North Carolina (USA), has a highly modular and flexible structure, and decomposes the problem of simulating crowds into component sub-problems, such as target selection, path planning, motion control, and collision avoidance. This separation allows users to easily modify or replace individual components, making it relatively easy to implement new models. Moreover, users can compare different algorithms by simply swapping modules. Menge is programmed in C++, offering high performance for large simulations.

However, while Menge is powerful, its graphic interface is very limited. It allows to visualize the movements of agents during the simulation, but the scenarios and parameters configuration is done via text files (Extensible Markup Language (XML)[2]), which is neither very visual nor intuitive. Indeed, users might find it complex to set up simulations. Menge also lacks built-in measurement tools, so users would need to develop their own analysis scripts.

4.1.2 JuPedSim

JuPedSim[3], developed at the Forschungszentrum Jülich (Germany), is focused on evacuation scenarios and safety studies. It includes several modules as JPScore for simulation management, JPSreport for output analysis, and JPSvis for visualization. It supports multiple pedestrian models, allowing users to study different behaviors. JuPedSim is mainly written in C++, with some additional tools developed in Python for data analysis and visualization.

One of JuPedSim's strengths is that it includes basic measurement tools capable of computing quantities like pedestrian flow, density, and velocity. On the other hand, adding new models into JuPedSim can be more challenging compared to other frameworks because of its less modular structure in terms of model extension. Indeed, even if JupedSim offers modularity by separating components like simulation (JPScore), analysis (JPSreport), and visualization (JPSvis), adding a new model within JPScore often requires deep modifications in the core C++ codebase. Finally, while JuPedSim provides visualization options, its user interface remains technical and is less accessible to new users.

[1] `https://gamma.cs.unc.edu/Menge/`
[2] `https://en.wikipedia.org/wiki/XML`
[3] `https://www.jupedsim.org/`

4.1.3 Vadere

Vadere[4], created by the Munich University of Applied Sciences (Germany), focuses on two-dimensional pedestrian dynamics and is highly user-friendly while maintaining high flexibility. It supports several locomotion models, including the four model we described in Chapter 2, and it allows to easily add new models into the framework without needing too much modifications. Vadere is programmed in Java, making it platform-independent: it can run on different operating systems (such as Windows, macOS, or Linux) without requiring big modifications. As a result, Vadere is easier to set up across different systems.

One of Vadere's biggest advantages is its intuitive graphical user interface, which simplifies scenario creation, simulation execution, and result analysis. Furthermore, Vadere integrates measurement tools directly within the software, making it easy to extract relevant data during or after a simulation. Vadere encourages contribution from the research community, making it a dynamic and evolving project. It also has a strong documentation which is a good point.

4.1.4 Choosing the best software

As a reminder, the main requirements stated earlier were flexibility, user-friendliness, and the possibility to easily integrate and compare different pedestrian models. Vadere seems to be the software that meets the highest number of requirements:

- It uses microscopic models to simulate individuals
- Our four models are implemented by default
- It supports easy addition of new models, with good documentation and a flexible structure
- It facilitates simple and direct comparisons between different models
- It provides built-in measurement tools for analyzing simulation outputs
- It offers an intuitive and user-friendly graphical interface
- Its platform-independence ensures easy deployment on different systems
- It is programmed in Java, which is a language that I master much more than C++
- Although C++ performs best in terms of execution time, Java is still effective compared to other programming languages (ex: Python) [Vis24]

[4] `https://www.vadere.org/`

This makes it a good option to be considered as the main framework in our future analysis.

4.2 Vadere's Architecture

4.2.1 Model-View-Controller Framework

Vadere's architecture is structured to provide extensibility, modularity and ease of use. It follows the Model-View-Controller (MVC)[5] pattern, dividing functionalities into distinct modules.

4.2.1.1 Model (`vadere-state`)

The Model represents the state of the simulation, and is referred as `State` in Vadere. It includes all the static and dynamic elements present in the simulation, such as the agents (pedestrians), obstacles, sources and targets. The Model defines the properties and behaviors of these elements, which are necessary for running simulations. The primary functions of the Model include:

- **Topography Elements**: agents, obstacles, and other objects in the environment are defined here. These elements interact with each other within the simulation.
- **Attributes**: the Model specifies the various attributes that define the agents' behavior, such as speed, direction, and personal space.

The Model is responsible for maintaining the simulation state and defining the logic behind pedestrian movements, but it does not handle the visualization or user interaction.

4.2.1.2 View (`vadere-gui`)

The View, referred as `Gui`, is responsible for the graphical user interface that allows users to interact with the simulation. It provides tools for scenario creation, simulation execution, and visualization of the results. Key components of the View include:

- **Scenario Editor**: this tool allows users to create and modify the simulation environment by placing pedestrians, obstacles, and targets within the scenario.

[5] `https://en.wikipedia.org/wiki/Model-view-controller`

- **Visualization Tools**: these tools display the simulation results in real-time, allowing to observe the pedestrians' movements and behaviors during the simulation.

The View interacts with the Model to retrieve the current state of the simulation and present it in an understandable format. The View ensures that users can interact with the system and monitor the simulations' progress.

4.2.1.3 Controller (`vadere-simulator`)

The Controller, or `Simulator` in Vadere, is the component that manages the overall simulation process. It controls the simulation loop, applies the selected locomotion model to update the pedestrians' positions, and interacts with both the Model and the View. The primary functions of the Controller are:

- **Simulation Loop**: the Controller manages the progression of the simulation, advancing the simulation step by step.
- **Behavior Models**: it is in this component that the various pedestrian models are implemented. These models define how agents move and interact with each other and their environment.
- **Measurement Tools**: it is also in this component that are implemented the measurements tools, i.e. several processors to retrieve significant data from the simulation, such as pedestrian flow, density or velocity.

The Controller serves as the link between the Model and the View, updating the simulation state and passing it on to the View for visualization. It also processes user interactions and modifies the simulation accordingly.

4.2.2 Interaction Between MVC Components

The three components of the MVC architecture interact with each other as follows:

- **Controller ↔ Model**: the Controller updates the Model's state by applying models and controlling the simulation progression. The Controller is responsible for computing changes to the simulation environment and the agents' states.
- **View ↔ Model**: the View retrieves the simulation state from the Model to visualize it for the user. The Model provides the necessary data to represent the simulation's current state.

- **Users ↔ View**: users interact with the simulation through the View, initiating actions that the Controller processes to update the model.

This clear separation of roles, where each component focuses on a specific responsibility, makes the Vadere framework highly modular and maintainable. This structure also allows to easily integrate new models or to modify existing ones without altering the main components of the system.

4.2.3 Supporting Modules

In addition to the three MVC components, Vadere includes several supporting modules that extend the functionality of the framework:

- `vadere-utils`: this module provides utility functions and classes for mathematical computations, geometry processing, and input/output operations. It is used across other modules to perform common tasks such as distance calculations or data parsing.
- `vadere-meshing`: this module provides tools for generating high-quality 2D meshes, which are used in the computation of floor fields and other spatial computations required in pedestrian simulations.
- `vadere-annotation`: this module allows users to annotate the simulation scenarios. This feature is useful for adding metadata or comments to specific elements, improving the scenario documentation and the collaboration among users.
- `vadere-manager`: this module is responsible for managing simulation projects and scenarios. It helps organize and maintain different projects, coordinates the interaction between various modules, and ensures smooth integration of the overall system.

4.3 Model Implementation in Vadere

One of Vadere's main strengths is its modular and extensible architecture, that allows to add new models easily. Every model must follow a standard structure by implementing a common Java interface called `Model`. This interface requires the implementation of four methods:

- `initialize()`: this method runs once at the beginning of the simulation. It sets up everything the model needs, such as calculating floor fields, initializing optimization methods, or preparing specific data structures.

- `preLoop()`: this method is called after the environment is set up but before the simulation actually starts. This step prepares any extra information the model might need right before the first simulation step.
- `update()`: this is the main method that is called at every time step of the simulation. It updates the state of each agent according to the model's rules, like moving pedestrians, calculating their interactions, or updating their destinations.
- `postLoop()`: when the simulation finishes, this method is called. It is used to clean up resources and save final data if necessary.

Integration of new models Thanks to this clear structure, new models could easily be added without needing to change the core parts of Vadere, just by implementing these four methods.

4.3.1 Simulation Loop

Vadere uses a time-stepped simulation loop to manage the movement of agents during the simulation. The simulation loop follows the same four phases than for the `Model` interface:

1. **Initialization**: the simulation environment is created based on scenario files. The `initialize()` method of the chosen model is called to set up all model-specific elements.
2. **Pre-Loop**: the `preLoop()` method is called. This step prepares anything that needs to be ready before starting the actual time-stepped updates.
3. **Main Simulation Loop**: for every time step:
   - The `update()` method of the chosen model is called.
   - Each pedestrian updates its position, speed and decisions according to the model's logic.
   - Special events, like reaching a goal or avoiding obstacles, are processed.
4. **Post-Loop**: when the simulation ends, the `postLoop()` method is called to finish recording data and release any used resources.

4.4 Graphic User Interface

In order to create and run simulations, Vadere provides a graphical user interface which allows to set up scenarios, configure simulation parameters, and run experiments visually. The GUI is designed to make the software accessible even to users who are not highly experienced in programming. With this GUI, users can easily create the environment, place pedestrians and set their goals.

Vadere uses structured JavaScript Object Notation (JSON)[6] files to store all the simulation parameters, that are called input files. The GUI allows the user to create these input files without explicitly writing the JSON file. It presents different tabs allowing to modify the simulation parameters directly in the JSON files or to change the topography in an intuitive interface without having to change the JSON file directly.

4.4.1 GUI Overview

The Vadere GUI is the main tool that users interact with when creating a simulation. It offers several features that make the process intuitive and efficient. The main window has three main parts:

- **Scenarios**: on the top left part of the interface there is a list of the created scenarios, and buttons to run these scenarios.
- **Output Files**: on the bottom left is displayed a list of the output files of the simulations.
- **Tabs**: the main part is on the right side of the window. It provides different tabs to create the scenarios and configure some parameters.

The main window is shown in Figure 4.1.

4.4.2 GUI Tabs

To create a simulation, we start by creating a new scenario. When this scenario is created, it is displayed in the list of scenarios on the top left side of the interface. We now have to configure all the parameters of this new scenario as creating the topography, choosing the simulation model or specifying the data we want to collect. We can do all that with the different tabs provided in the main part of the interface. With each tab comes a specific JSON file.

[6] `https://en.wikipedia.org/wiki/JSON`

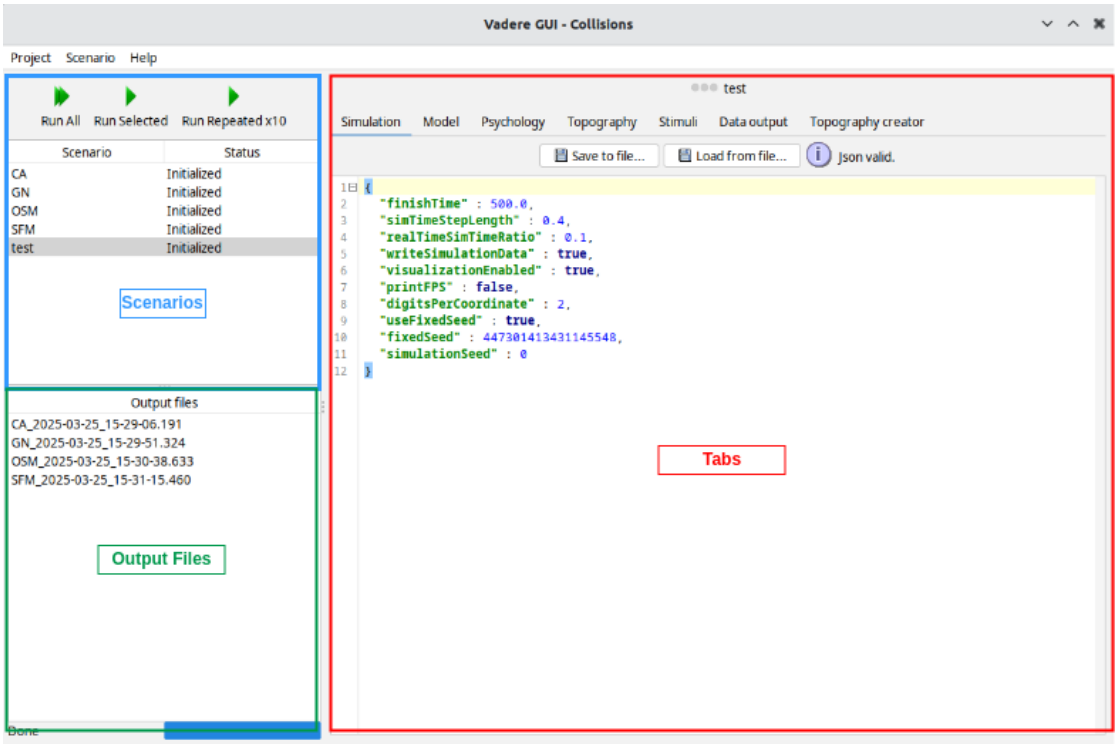


Figure 4.1: Graphical User Interface

4.4.2.1 Simulation Tab

The first tab contains the JSON file associated with the parameters of the simulation, as the maximal finish time in second of a simulation, or the time step length. Users can change these parameters in the JSON file directly with the interface.

4.4.2.2 Model Tab

The second tab is the tab where the user can specify the mathematical model he/she wants to use so to run the simulation. One can find a list of models to upload with their JSON files already written, and then modify the model parameters at will.

4.4.2.3 Psychology and Stimuli Tabs

The psychology tab is used to specify any psychological aspect users would want to add to the simulation. They can decide here if they want to include the psychological aspect to simulations or not. It also defines a priority system for pedestrian behaviors in order to resolve potential conflicts due to the multiple stimuli received at the same time. The default priorities are the following, with highest priority corresponding to the lowest numerical value:

1. **InformationStimulus**: pedestrians react to new information, that are encoded in the Stimuli Tab.
2. **ChangeTargetScripted**: pedestrians follow a predefined script to change their target.
3. **ChangeTarget**: pedestrians decide to change their target themselves based on dynamic environmental factors, such as congestion or obstacles.
4. **Threat**: pedestrians perceive a threat (ex: explosion, fire, ...) and react by changing their path to avoid danger.
5. **Wait**: pedestrians decide to wait, due to obstructions or waiting for others.
6. **WaitInArea**: pedestrians wait within a delimited area.
7. **DistanceRecommendation**: pedestrians adjust their path to maintain recommended distance from each other.

By default, the use of the psychological layer is set to false.

4.4.2.4 Data output Tab

The data output tab is a very important one, because it allows users to choose which data they want to save while running a simulation. The tab provides an intuitive interface to add some data processors, which will save the chosen data, and some output files to store these data. Users can choose among a various list of data processors, and they can decide to store a lot of different measurements as the position of pedestrians at each time step, their speed at each time step, the density of a specific area, or even the evacuation time of the simulation. One can also find in this part the velocity, flow and density measured with the four different methods for the fundamental diagram, as discussed in Section 3.1.3.

4.4.2.5 Topography and Topography creator Tabs

These two tabs are the most important ones, as they are used to create the topography of the scenarios. They are highly linked, because the Topography creator tab allows to intuitively create the topography by dragging elements into it, and the Topography tab contains the JSON file that is automatically updated as elements are added to the topography. The Topography creator tab presents as shown in Figure 4.2.

In the right side, there are three little tabs that give information about the topography:

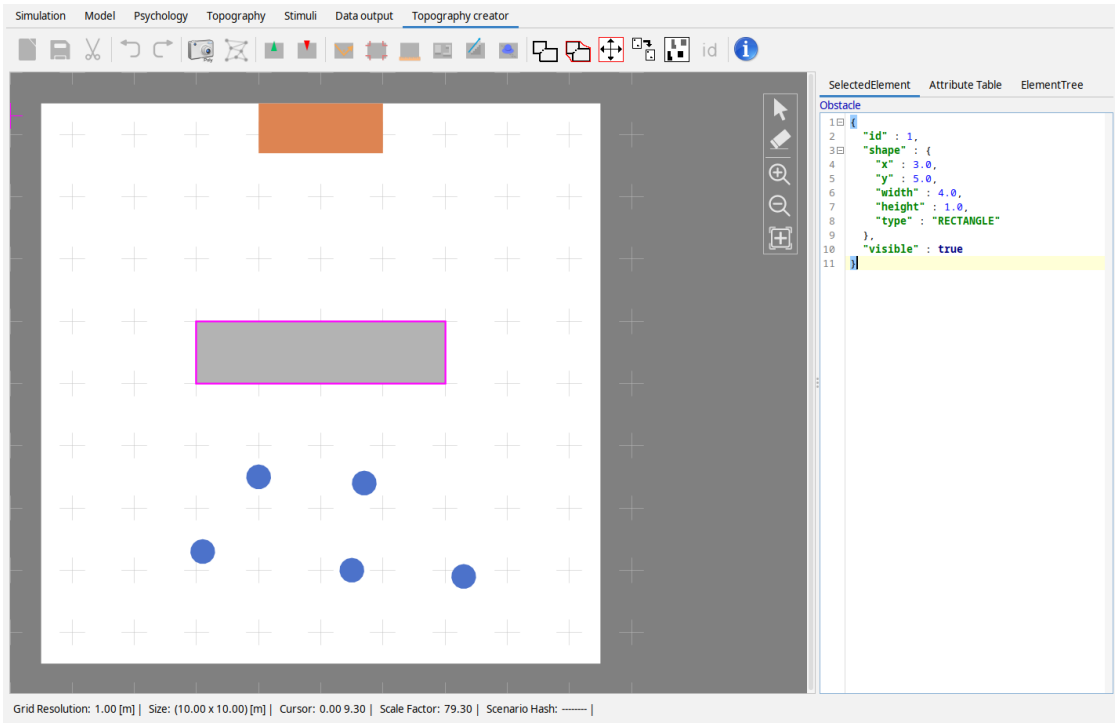


Figure 4.2: Topography creator Tab

- **SelectedElement**: gives the part of the JSON file corresponding to the selected element. Users can find all sorts of information about it, as its id, its position or its size. If they want to modify the selected element, they can either do it with the mouse on the topography, or change it directly in the JSON file.
- **Attribute Table**: also gives information about the selected element, but with a table, not the JSON file. One can also modify the element attributes in this table.
- **ElementTree**: gives a list of all elements in the topography with their id.

In the main part, users can drag all sort of elements they want to create a scenario, and also change the bounds of the topography. Here are all the elements one can add to the topography:

- **Source**: Area from where pedestrians spawn into the simulation. The entry rate of pedestrians and their initial properties can be controlled.
- **Target**: Destination area pedestrians aim to reach.

- **Target Changer**: Area that changes the pedestrian's destination (target) when the pedestrian enters it.
- **Absorbing Area**: Area where pedestrians are removed from the simulation. It represents endpoints of the simulation, such as exits or destinations.
- **Measurement Area**: Area used to measure some metrics within the specific area. One can for example measure the density of a certain area during the simulation.
- **Obstacle**: Area that the pedestrians have to avoid.
- **Stairs**: Bloc representing staircases.
- **Pedestrian**

When creating pedestrians, several choices are presented. Users can either create them one by one by hand, and manually set the target they have to reach, or pedestrians can be placed at random in a delimited zone. In this case, the number of pedestrians, the zone delimitation, and how to assign a target to pedestrians have to be specified. There are three ways to assign a target to a pedestrian: either not assigning a target to the pedestrian, assigning a target at random, or user can encode a list of targets among which the target will be randomly chosen. Other parameters than their assigned target can be specified, as their desired walking speed or their body radius.

4.4.3 Visualization of Simulations

Another important part of the GUI is the real-time visualization of the simulation. Once the simulation starts, users can see how agents move through the environment, avoid obstacles, and interact with each other. Once the simulation is finished, we can replay the simulation frame by frame to analyze the simulation afterwards. We can also add some visual information, such as the pedestrians' trajectories or their walking direction.

4.5 My Software Modifications

I added some functionalities to the software in order to simplify the creation of scenarios.

4.5.0.1 Information Button

At the request of the firefighter we were in contact with, I added an information button in the GIU, in order to have several information about the created topography. The important information to collect for the firefighter were the number of exits, the total length of exits, because they have standards that impose a minimum length of emergency exit depending on the number of people to evacuate, and the zone in the space that is the furthest from any exit. This button then displays the total number of pedestrians in the topography, the number of exits (corresponding to the targets), the total length of exits, the furthest point from any exit and finally the distance separating this point and the nearest exit.

In order to compute the point of the topography the most far away from any exit, I used a Voronoi diagram-based method. A Voronoi diagram is *a partition of a plane into regions close to each of a given set of objects*[7]. In other words, each cell in the diagram contains the points that are closest to a particular site. In order to find the furthest point from any site, the algorithm iterates over the vertices of the Voronoi diagram and computes the minimal distance between this vertex and the closest site. Then, the furthest point from any site will be the vertex with the maximal minimal distance. An example of diagram is shown in Figure 4.3.

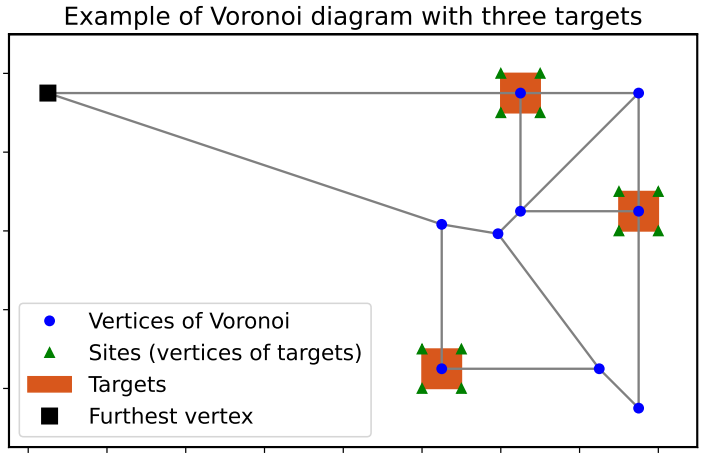


Figure 4.3: Voronoi diagram for an example with three targets

In our case, I used the vertices of all targets as the set of sites to generate the corresponding Voronoi diagram.

The added button and the pop up are shown in Figure 4.4.

[7] `https://en.wikipedia.org/wiki/Voronoi_diagram`

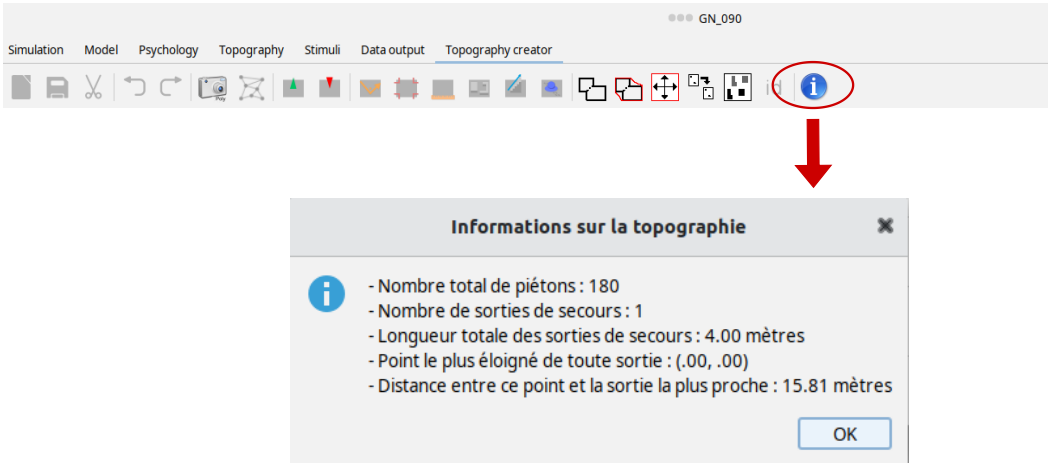


Figure 4.4: Pop up and button added to display information about the topography

4.5.0.2 Nearest Target

When creating pedestrians at random in the topography, there were three propositions for assigning the desired target to the pedestrians: assigning nothing, assigning the target at random or specifying a list of targets. I added an option: assigning to each pedestrian its nearest target as this is the most natural way to choose the target we want to reach. The resulting tab is shown in Figure 4.5.

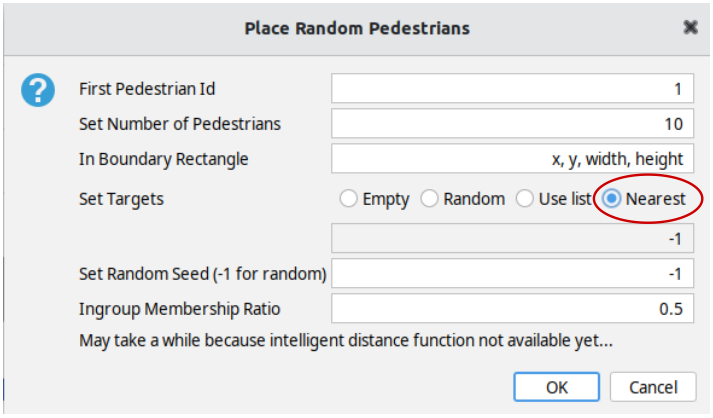


Figure 4.5: Adding the option to assign the nearest target to each pedestrian

4.5.0.3 Add Pedestrian Zone

In addition to placing a chosen number of pedestrians at random in a delimited zone, I added an option: drawing a zone on the topography and imposing a pedestrian density in the designated zone. After selecting the option *Place Pedestrian Density Zone*, the user can draw the zone with the mouse, finishing by a double click. After that, a tab opens when the user can specify the id of the first pedestrian, the desired density, and choose how to assign the target to pedestrians. When the user has validated his/her choices, a tab pops up with information about the created pedestrians: the area of the designated zone, the desired density, the actual density (the desired density cannot always be obtained because we cannot add half a pedestrian) and the number of added pedestrians. This is illustrated in Figure 4.6.

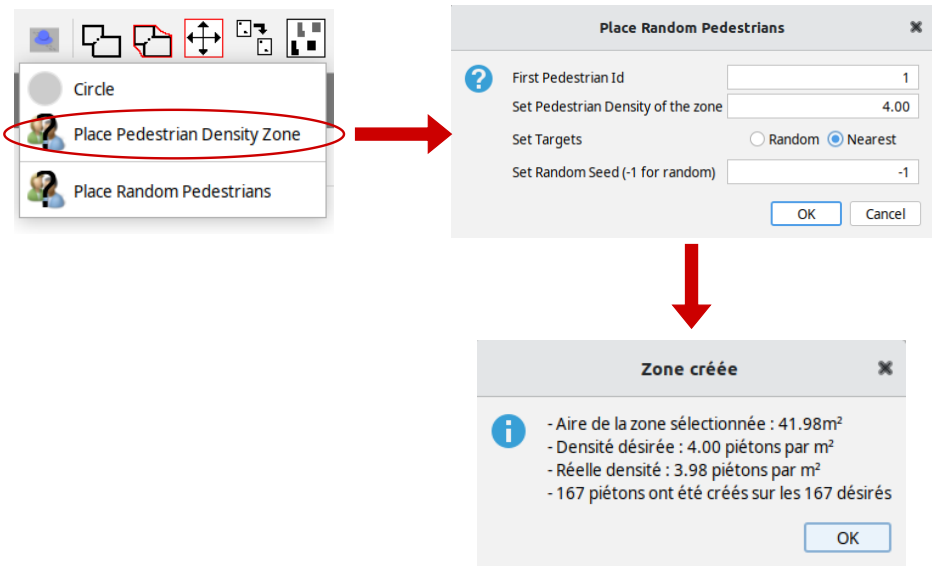


Figure 4.6: Option to place a pedestrian density zone

I found that important to add because when talking with the firefighter we were in contact with, he talked about pedestrian density zones, and said that it could be interesting if we were able to impose a certain density of pedestrians in certain zones.

Experimental Part 5

After presenting the mathematical foundations of the four crowd simulation models, the different comparison metrics, and the simulation software used, this section focuses on the experimental evaluation of the models. The goal is to identify which model performs best in our practical case which is the *24h vélo of Louvain-la-Neuve*.

In order to do this, we have run several simulations using each model and compared them based on specific criteria, including adaptability to different scenarios, stability, computational time and realism of crowd movement. These criteria were selected to reflect the real needs of professionals working in crowd management and emergency response.

To start with, we chose to use the default parameters of the models described in Section 2 for simplicity. Each model includes many parameters that could be adjusted, but testing all of them would be too time-consuming. Therefore, we first run the simulations using the default settings. If one or two models show promising results for our specific case, we will then explore in a second section how varying their parameters can improve their performance.

The first section will describe the scenarios used for each simulation, present the results, and discuss the strengths and weaknesses of each model. The second section will do the same but while comparing different set of parameters instead of comparing different models. If a model shows promising results, we will try to compare it with real data in the last section. The results of all the experiments will help determine which model is the most suitable for operational use, and with which parameters.

5.1 Model Comparison

Pedestrian Parameters We have to determine a set of parameters for the pedestrians. The two most important parameters we will have to find are the radius of the torso of one pedestrian, as it is used in lots of models as the OSM and the CA, and the free-flow speed, used in all four of our described models. We take 0.2m for the radius of a pedestrian's torso, the default value set in Vadere. For the free-flow speed, we will take the results of the empirical meta-study of Weidmann [Wei93] and assign to each pedestrian a free-flow speed normally distributed around mean 1.34m/s and standard deviation 0.26m/s. We set a minimum speed of 0.5m/s

and a maximum speed of 2.2m/s, values also found in the work of Weidmann.

We can vary other parameters, as set the individual as a child or not, or make the pedestrian part of a group of pedestrians that will stay together during the evacuation. Vadere also proposes a psychology layer that can influence the decisions of an individual, but that will not be taken into account here, for simplicity.

Note that in an event such as the *24h vélo of Louvain-la-Neuve*, the presence of alcohol is an important factor to take into account. Indeed, drunk people do not move in the same way that sober people. Someone's behavior in face of an emergency can also be altered by alcohol ([Dul+11]). For now, our models do not take the presence of alcohol into account. We have not found any study that evaluates the influence of alcohol on the walking speed of pedestrians, but if such a study is conducted in the future, we could adapt the free-flow speed of pedestrians accordingly.

5.1.1 Do pedestrians always reach their goal?

One of the first questions to ask ourselves when trying to determinate which model we are going to use is whether this model leads all the pedestrians to their targets, or if it fails at this task. For that, we designed some tricky scenarios, and we observed for each of them if all pedestrians reach their targets or if some of them get stuck behind obstacles or other stuff.

Description of chosen scenarios We created 8 different scenarios that represent simple and more complex real-life situations, like a simple street or big places to evacuate. A brief description of the scenarios is given below and a scheme of each of them is available in Appendix A.1.

- **Scenario 1**: A simple street. 10 pedestrians are placed at one end of a street and a single target at the other end. There is no obstacle in the way. It is the simplest scenario.
- **Scenario 2**: The same street as before but with stairs in the middle of the way. We also simulated 10 pedestrians.
- **Scenario 3**: Simple scenario where 30 pedestrians have to bypass an obstacle to reach a single target placed behind it.
- **Scenario 4**: A street without any obstacle but in *zigzag*. As the street is narrow, we only simulate 5 pedestrians.
- **Scenario 5**: A bottleneck situation, where 30 pedestrians have to walk through a narrow passage between two obstacles to reach the single target.

- **Scenario 6**: A big place with 400 people to evacuate through 2 exits.
- **Scenario 7**: A big place with 400 people to evacuate through multiple (5) exits.
- **Scenario 8**: A big concert place that looks like the *Parking Leclercq* (one of the biggest parking of the event *24h vélo*) with a thousand of pedestrians to evacuate through 3 emergency exits.

We created these scenarios based on several criteria such as the presence of obstacles, stairs, the ability to simulate change of direction, the ability to evacuate through several exits, and the ability to simulate large crowd density. We represented in Table 5.1 which scenario is generated to test which criteria.

Scenario n°	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Stairs		×						
Obstacles			×		×	×		×
Change of direction				×		×		
Multiple exits						×	×	×
Large crowd density						×	×	×

Table 5.1: Description of the testing criteria for each scenario

We ran all these scenarios with our four different models and we checked if pedestrians managed to reach their target. We computed the percentage of the initial pedestrians that successfully arrived to their target and gathered the results in Table 5.2.

Scenarios	**OSM**	**CA**	**SFM**	**GNM**
Scenario 1	100%	100%	100%	100%
Scenario 2	100%	100%	100%	100%
Scenario 3	100%	83.33%	100%	100%
Scenario 4	100%	0%	100%	100%
Scenario 5	100%	63.33%	100%	100%
Scenario 6	100%	15.75%	TimeOut	100%
Scenario 7	100%	66.5%	100%	100%
Scenario 8	100%	73.6%	TimeOut	100%

Table 5.2: Percentage of pedestrians that reach their target for each scenario simulated with each model

For the Optimal Step Model and the Gradient Navigation Model, there is no problem, every pedestrian in every scenario manages to reach its goal. For the

Cellular Automaton, we encounter a problem as soon as obstacles are injected in the topography. The pedestrians get stuck behind obstacles or each other, and do not reach their target. The simulation then waits until the maximum simulation time and ends. And for the Social Force Model, all pedestrians usually reach their target, but for some configurations the execution time for the simulation was too long ($>$ 10 minutes). It happened with a big number of pedestrians, but this will be discussed later.

5.1.2 Are the simulated trajectories realistic?

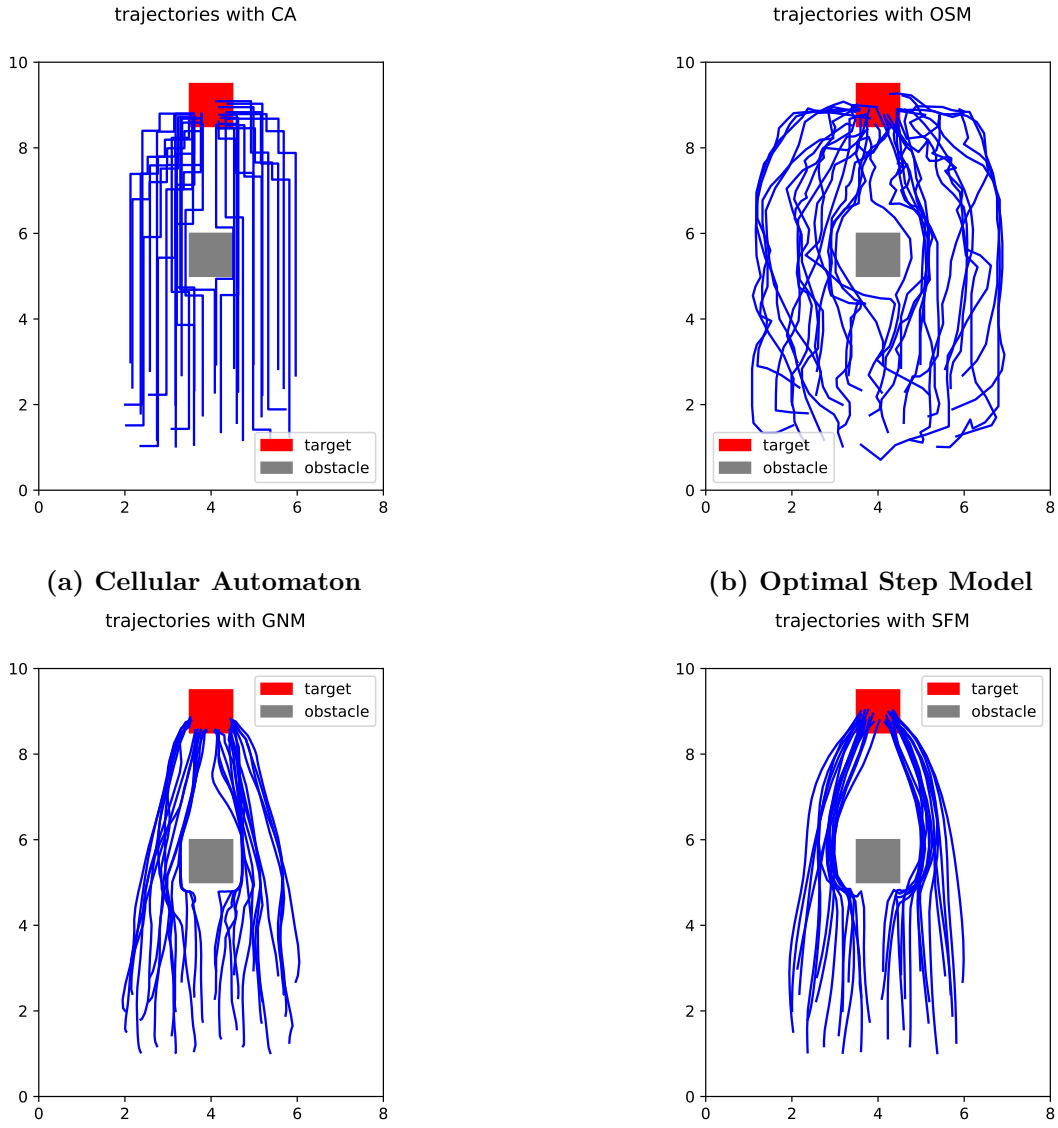
Although it is a subjective metric, the trajectory analysis is an important point in pedestrian simulations. Here we just want to take a look at the trajectories described by our four models in a simple scenario, and see if we can conclude some interesting things or not. The scenario is very simple: a group of 30 pedestrians walking toward a target, with an obstacle in their way. The trajectories are plotted in blue in Figure 5.1, the target in red and the obstacle in gray.

We can clearly see that there are differences between the simulated trajectories. Some models simulate smooth trajectories while others indicate more disordered paths.

For the Cellular Automata, we can clearly observe the passage from cell to cell of the algorithm, as the trajectories are very straights and there are right-angle changes of direction. Such a trajectory is due to the discretization of the space, and is not very realistic. We could try to improve the trajectories by defining smaller size of cells. For the Optimal Step Model, trajectories are very disordered and pedestrians seem to move away from each other more than in the other models. For the Gradient Navigation Model and the Social Force Model, the trajectories seem smoother and are more likely to match the reality.

5.1.3 Which model is the most invariant to the random placement of pedestrians?

Let's take a look at the stability of the evacuation time by investigating how random variations of the initial scenario influence the time of evacuation predicted by our models. This is an important metric because pedestrians will probably be placed randomly in the topography as there can be thousands of them in a place to evacuate and it would be too tedious to place each of them by hand. Furthermore, in big crowded places, people tend to move and not stay at their initial position, so it would be impossible to determine an exact position for every pedestrian. As the evacuation time is an important measure for the firefighter and other emergency authorities, we want it to be the most exact possible and stable through the random positioning of pedestrians.



(a) Cellular Automaton **(b) Optimal Step Model**

(c) Gradient Navigation Model **(d) Social Force Model**

Figure 5.1: Trajectories of pedestrians

Choice of scenario In order to determine the stability of the evacuation time, we took firstly the most basic evacuation scenario, which is the evacuation of a place towards an emergency exit. We have to take a number of pedestrians that is not to little, because we need some interactions between them, but not too big for the computation time. We chose to simulate 100 pedestrians, in a $30m^2$ square, so the density inside the square is 3.33ped/m^2 which is a good compromise. A schema

of the topography can be found in Appendix A.2.

Results We decided to run the simulation 100 times with different random configurations of pedestrians, and observe the distribution of the evacuation time. We noticed that for the Cellular Automaton, the evacuation time was always 499.9s, which is the maximal time for a simulation. This means that some pedestrians got stuck and did not reach their targets, so the simulation only stopped at the end of the maximum simulation time and the evacuation time is then not relevant.

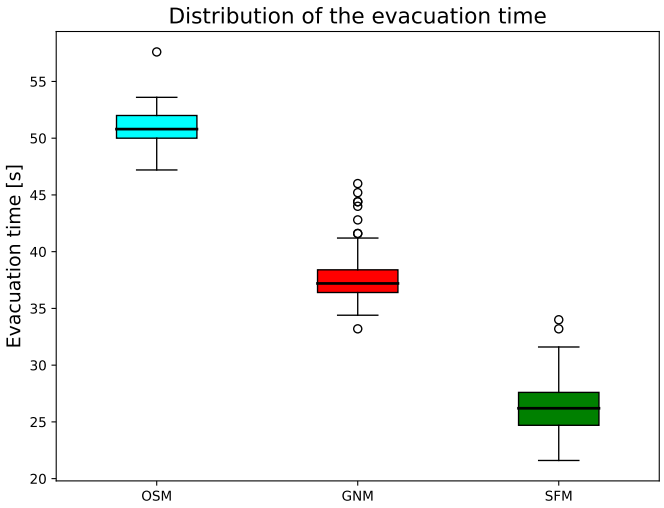


Figure 5.2: Distribution of the evacuation time

In the graph shown in Figure 5.2, the box represents the middle 50% of the values (called the interquartile range), and the line inside the box shows the median, which is the middle value. The "whiskers" on either side of the box show the range of values that are not considered outliers, while the small circles outside the whiskers represent outliers, which are values that are much higher or lower than most of the others.

We can observe firstly that the evacuation time differs according to the models, but without any experimental validation we cannot determine which one is the most realistic. Regarding the stability, the SFM model shows the largest variation in evacuation times. Its box is the widest and its whiskers are relatively long, indicating that results differ more from one pedestrian setup to another. It also has a few outliers, suggesting some unusually long evacuation times in certain cases. The GNM model has a narrower box, meaning that the evacuation times are more tightly grouped, but it has many outliers above the box, showing that

longer evacuation times still occur occasionally. The OSM model, while having a fairly compact box, includes a high outlier and slightly longer whiskers, showing moderate variation in its results.

5.1.4 How well do the models simulate the avoidance of collisions?

The next metric to analyze is the collision avoidance. We know that pedestrians tend to stand and wait instead of bumping into each other. This is basic human behavior, and we want to know how well our models replicate this behavior. We measured the total number of collisions (TC) and the average number of collisions per time step (AC) in a bottleneck scenario of 300 pedestrians, because it is a scenario that pushes pedestrians towards the same place. We computed these metrics by taking the positions of each pedestrian at each time step, and then counting at each time step the number of pairs of pedestrians that are in collisions, i.e. the distance between their centers is less than 0.4m, as the radius of one torso is 0.2m. The results for each model are gathered in Table 5.5.

Models	**TC**	**AC**
OSM	408	1.83
CA	35766	109.04
SFM	11931	135.58
GNM	4413	20.82

Table 5.3: Number of collisions

We see here that the Optimal Step Model performs the best in terms of collision avoidance, both for the total number of collisions and the average number of collision per time step. The Gradient Navigation Model comes at the second place, but has ten times more collisions that the OSM for the same scenario, which is significant. The Social Force Model and the Cellular Automaton Model have an important number of collisions.

5.1.5 Which models handles the best a growing number of pedestrians?

We want to study in this section the ability of our models to handle an increasing number of pedestrians. We will look into the main measure: the execution time of one simulation, because we want the model to simulate a scenario in a reasonable amount of time.

Choice of scenario For the beginning, we chose the simplest scenario in order to test simply if our models can simulate some pedestrians walking towards a target.

Our topography corresponds to a $17m \times 28m$ rectangle with a target on one side, and the pedestrians are placed within a $15m \times 15m$ square on the opposite side of the target, a scheme of the topography is shown in Appendix A.3.

After doing that with the simple scenario, we tested the same amounts of pedestrians on a more complex topography, the same as before but we added some obstacles. We want to investigate the influence of the presence of obstacles on the execution time, because in the real life it will not be a simple scenario with no obstacles, and we still want our model to be effective. A schema of the new topography is also available in Appendix A.3.

Execution Environment and System Configuration The simulations were carried out on a machine equipped with an 11th Gen Intel Core i7-1165G7 processor (4 physical cores, 8 threads, base frequency 2.80 GHz, turbo up to 4.70 GHz), 15 GiB of RAM, and an integrated Intel Iris Xe Graphics (TigerLake-LP GT2) GPU. The system was running Ubuntu 22.04.5 LTS with Linux kernel version 6.8.0-60-generic, on a 64-bit x86_64 architecture.

Results We ran simulations firstly with 10, 50, 100, 250, 500 and 750 pedestrians randomly placed in the pedestrian area, and we recorded the execution time. We showed what we obtain for the first simple scenario in Figure 5.3 and for the more complex scenario in Figure 5.4.

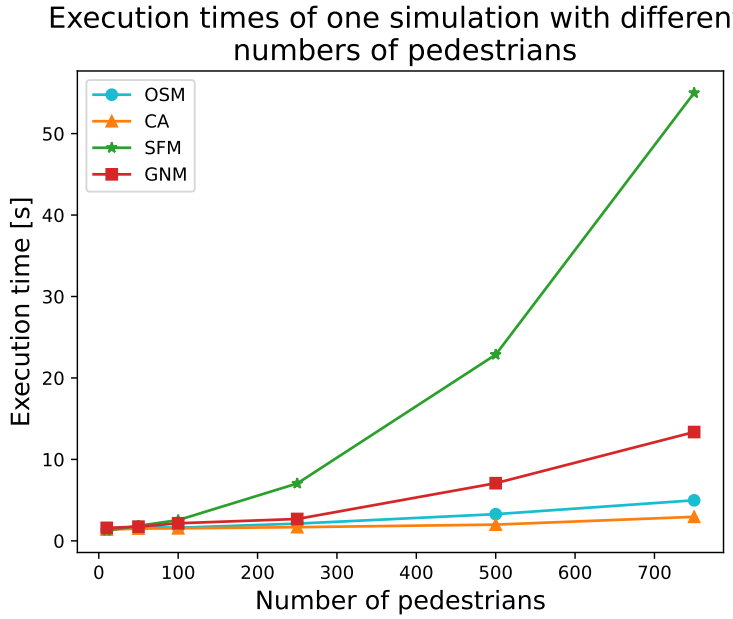


Figure 5.3: Execution time of the simple scenario

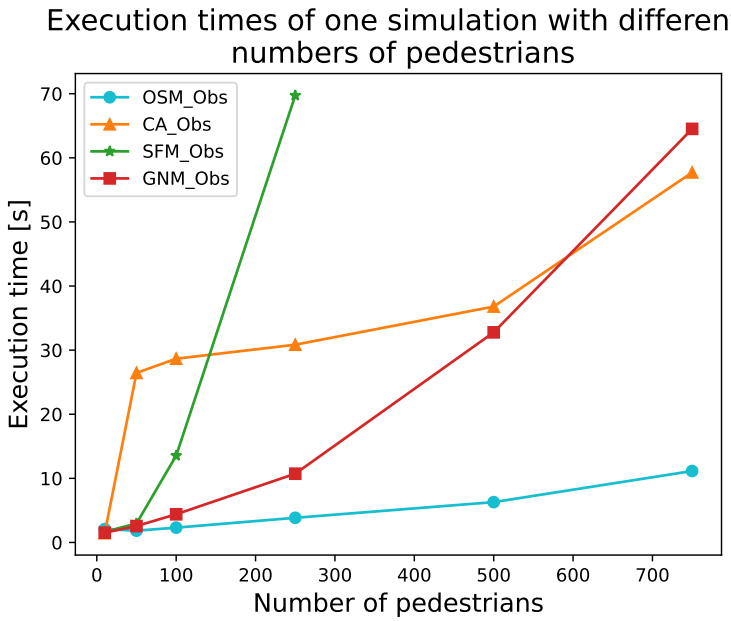


Figure 5.4: Execution time of the complex scenario, with obstacles

We can see that the execution time grows with the number of pedestrians, which is rather logical. We observe that the Social Force Model is the most time consuming model, with more than 50 seconds for 750 pedestrians in the simple scenario, and almost 10 minutes for 500 pedestrians in the complex scenario. We cut off the SFM curve at 250 pedestrians for it to fit in the graph.

We also see that adding obstacles to the topography increases the execution time, which is also natural as adding more elements to the topography increases the computations. The model that keeps the best time performance with growing number of pedestrians is the Optimal Step Model, even with the obstacles. The Gradient Navigation Model is not bad either, but we see that adding obstacles multiplies the execution time by more than 2.5. We can explain the shape of the curve for the Cellular Automata in the complex scenario by the fact that for a very little number of pedestrians (under 50), the model can normally simulate the evacuation. But for more pedestrians, they get stuck behind obstacles and they are not able to reach their assigned target, so the simulations runs until the maximal simulation time. That is why we observe a gap between the execution time of 10 pedestrians, and the execution time of 50 and more pedestrians.

As our goal is to simulate the evacuation of large places, we tried with even more pedestrians for the Cellular Automata, the Gradient Navigation Model and for the Optimal Step Model, to see how big they can take. We tested them on the simple scenario, without obstacles and reported the results in a graph on Figure

5.5.

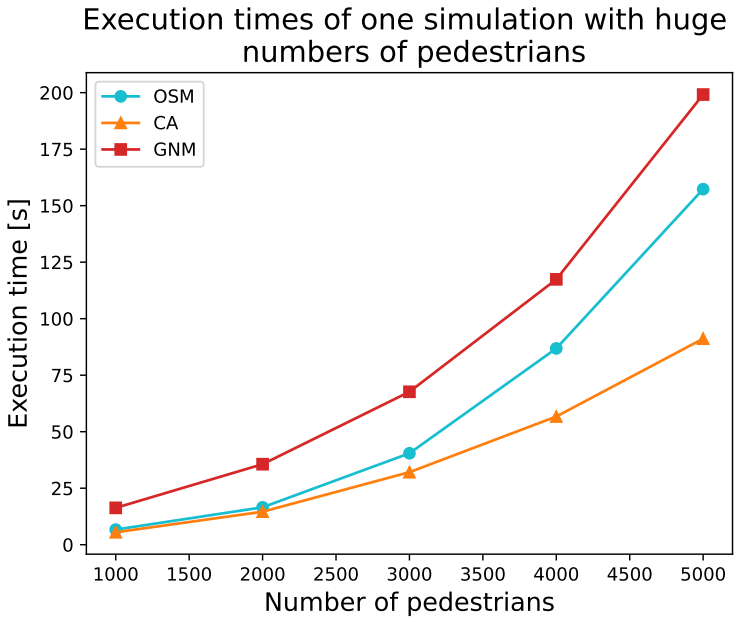


Figure 5.5: Execution time according to the number of pedestrians

We can see that these three models can simulate crowds of thousands of pedestrians in a reasonable amount of time. We know that these times would increase in more complex scenarios, with more obstacles.

5.1.6 Fundamental Diagram

The last aspect we wanted to investigate is the relation between the velocity or flow and the density, i.e. the fundamental diagram. In order to have significant data, we had to create a significant scenario where the density varies with time, and also where we could find a significant measurement area. The method used to gather the velocities and densities will also be discussed.

As seen in several articles ([Lia+14], [KGS06], [Lid+11], [Sey+09]), the velocity-density relation is often studied in bottlenecks scenarios, because it is the simplest scenario where we can observe a significant change of density and velocities of pedestrians as they enter the bottleneck. We chose a similar scenario, and we used the same measurement areas than [Lia+14], i.e. two different measurement areas: one in the bottleneck (Area 1) and one just in front of the bottleneck (Area 2), a scheme of the topography is showed in Appendix A.4.

We used the methods B, C and D discussed in Section 3.1.3 to measure the density of the measurement area and the average velocity, and used them to compute

the flow. We did not test the method A because it does not use a measurement area but a line and in order to compare the different methods, it is important that the measurement areas are the same in all methods. We did not use the Cellular Automaton model in this part because once again, the pedestrians get stuck and do not always reach the target, which would give insignificant data.

We simulated the scenario with several bottleneck lengths $b_{cor} = 0.9\text{m}, 1.5\text{m}, 2\text{m}$ in order to collect more data and to analyze the effect of the bottleneck length on the measured metrics. All graphs for all models, measurement areas, and measurement methods are available in the Appendix B.

5.1.6.1 Discussion about the measurement method

As a reminder of Section 3.1.3, the three methods measure the velocity and the density over a rectangular measurement area, but in three different ways.

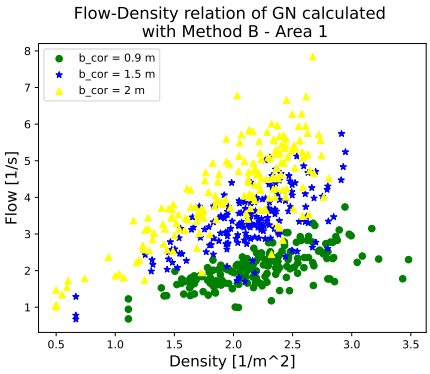
The **Method B** measures the velocity and average density of each pedestrian passing through the measurement area. The velocity is simply computed by taking the length of the measurement area divided by the time taken to cross it, and the density is the integral within the time the pedestrian enters the measurement area and the time he exits it of the total number of pedestrians in the measurement area, divided by its area and the time taken by the pedestrian to walk through the measurement area. In the graphs, each point corresponds to one pedestrian.

The **Method C** computes at each time step the number of pedestrians in the measurement area, and divide it by the area of the measurement area to obtain the density. Then, it measures the average of the instantaneous velocities of the pedestrians in the measurement area at this time step. In the graphs, each point corresponds to one time step.

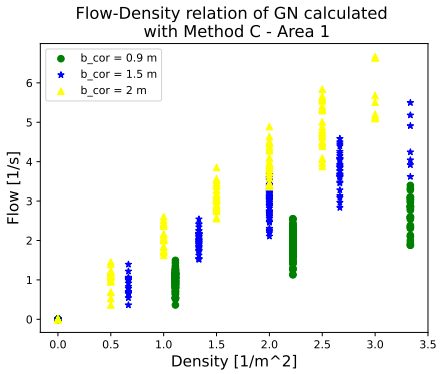
The **Method D** also computes at each time step the density and the velocity but using the Voronoi cells. The density is equal to the sum of areas of all Voronoi cells intersecting the measurement area divided by its area. The velocity of a pedestrian is defined by its velocity multiplied by the area of its Voronoi cell. Therefore the average velocity is the sum of all those pedestrian velocities of Voronoi cells intersecting the measurement area divided by its area. In the graphs, each point corresponds to one time step.

Let's take for example the flow-density diagrams for one of our models, the Gradient Navigation Model, and for the first area. The diagrams are shown in Figure 5.6.

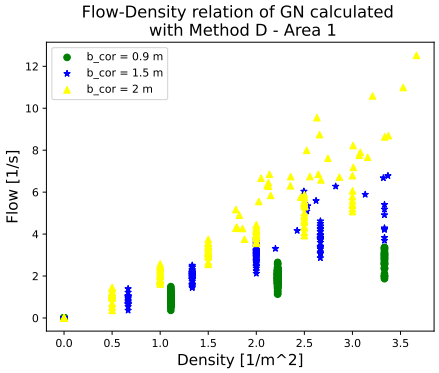
We observe the same tendency in the three methods: the flow grows with the density, and we have higher flow values for wider bottleneck, which is rather normal as more people can pass through a wider passageway. We see that for method D, the flow values are a little higher than the two other methods, and so are the velocities values (see graphs in Appendix B). We could explain that by the fact



(a) Method B



(b) Method C



(c) Method D

Figure 5.6: Flow-density relation for GN for different methods

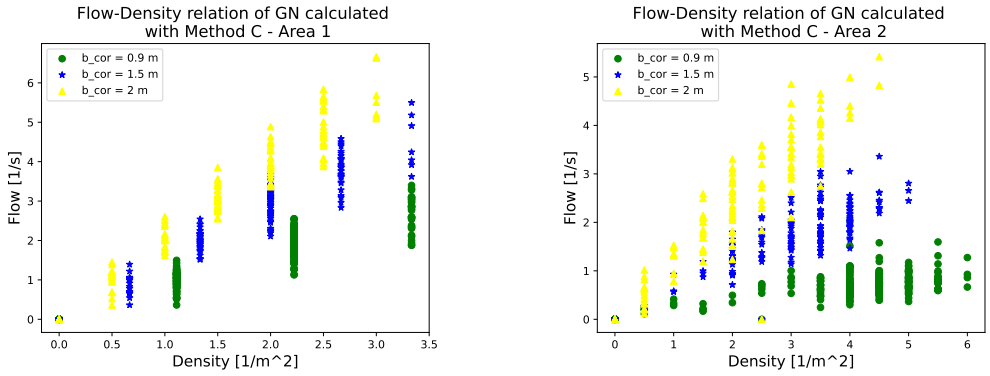
that Method D looks at each pedestrian's speed at a specific moment and in their own space, which makes it more precise. Methods B and C, on the other hand, calculate average speeds over time or over a group of pedestrians, which can make the results lower. That's why Method D often gives higher velocity values.

5.1.6.2 Discussion about the measurement area

Two different measurement areas are analyzed: one in the bottleneck (Area 1) and one just in front of it (Area 2), to see which zone is more congested.

We took for example the fundamental diagram of the Gradient Navigation Model computed with Method C. The graphs are shown in Figure 5.7.

The density for the first area varies between 0 and 3.5 pedestrians/m^2, while in the second area, just in front of the bottleneck, the density can increase until 6 pedestrians/m^2. And this is a general tendency in other models and methods, see Appendix B. That means that if we want to collect significant data about the congestion of a zone, it is better to observe the zone just before the bottleneck



(a) Area 1 **(b) Area 2**

Figure 5.7: Flow-density relation for GN with different measurement areas

than the bottleneck itself.

We can also observe that the flow increases with the width of the bottleneck which is an expected behavior.

5.1.6.3 Discussion about the model

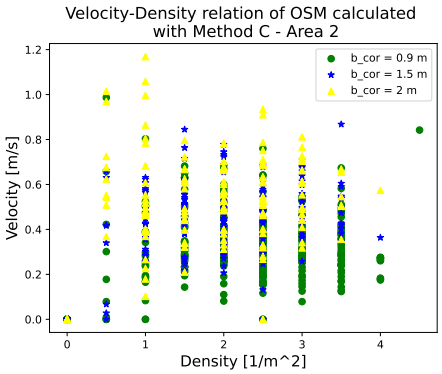
In a normal speed-density relation, the speed should decreases as the density increases. We will check in this section if our models rightly simulate that or if we see some weird behaviors. Let us take a look at the second area while using method C to measure the data in Figure 5.8.

The first thing we notice from these three graphs is that in the Social Force Model, the density can go up to 15 pedestrians per square meter, which is far from realistic. In real-life situations, the maximum density in crowded places is usually around 6 pedestrians per square meter. We could explain that by the fact that the model doesn't take into account the personal space people need, and it ends up simulating them pushing or overlapping each other.

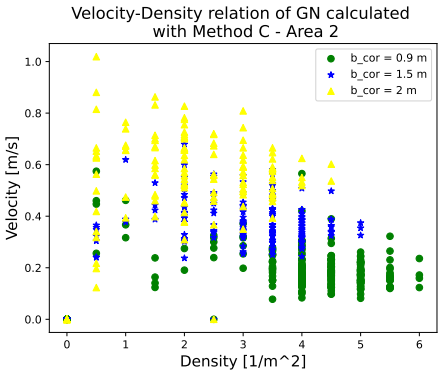
For the two other models, the Optimal Step Model and the Gradient Navigation Model, it is difficult to say if the velocity decreases when the density increases, but we can say that it does not increase. The data collected for the three different bottleneck width are more scattered for the Optimal Step Model than for the Gradient Navigation Model.

5.1.7 Summary

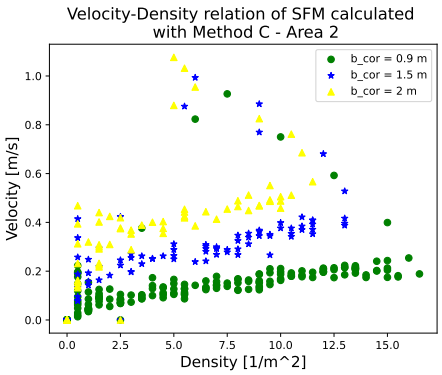
We have studied several metrics in order to evaluate and compare our models. It is time to see which model we want to keep and which one we will eliminate. A summary of the results of each model for each metric is done in Table 5.4, with



(a) Optimal Step Model



(b) Gradient Navigation Model



(c) Social Force Model

Figure 5.8: Velocity-density relations for the different models

the following legend:

★ : Very good

✓ : Good

~ : Fair

✗ : Poor

– : Inconclusive results

Let's begin with the **Cellular Automaton**. We have seen with the first set of simulations that this model does not work in every situation, because pedestrians do not always reach their goal. This is a prohibitive criteria, because we cannot measure anything with a model that does not even finish the simulation correctly. Moreover, the simulated trajectories do not seem realistic, with straight line moves. This suffices to rule out this model from the competition.

	OSM	CA	SFM	GNM
Do pedestrians always reach their goal?	★	✗	✓	★
Are the simulated trajectories realistic?	~	✗	★	★
Which model is the most invariant to the random placement of pedestrians?	✓	–	✓	✓
How well do the models simulate the avoidance of collisions?	★	✗	✗	~
Which models handles the best a growing number of pedestrians?	★	–	✗	✓
Fundamental Diagram	✓	–	✗	✓

Table 5.4: Summary of simulation results

The **Social Force Model** would be a good model for low-scale scenarios. Indeed, the trajectories seem really smooth, and the small scenarios in the first simulations where handled correctly. But the main problem with this model is the computational cost. A simulation of a scenario with more than 500 people becomes really long to execute, and this is a big problem when we want to simulate places with thousands of people. Moreover, the collision avoidance is not well handled and the model sometimes simulates zones with a density of 15 pedestrians per square meter, which is impossible in real life. We will therefore also remove this model from consideration.

The two last models, the **Optimal Step Model** and the **Gradient Navigation Model** both have their strengths. They both can simulate many different scenarios without any visible problem and they are both pretty fast to simulate large numbers of pedestrians, even if the OSM is slightly faster than the GNM. On the other hand, the GNM performs best in term of realism of the trajectories, which is an important point. There is no visible problem with their fundamental diagrams, and they are more or less stable through random placement of pedestrians. We can give an additional point to the OSM for the collision avoidance, because it performs better than GNM in this field. But we can say that these two models seem good for large-scale simulations. We decided to keep both of them for the parameter variation, and we will see in the next section if one of them can be chosen as the best model.

5.2 Parameters Variation

Now that we have found two good candidates to run our simulations, we can try to improve them by playing with their parameters. We will test some metrics

that we find interesting for some parameters, and investigate the influence of these parameters on the chosen metrics.

5.2.1 Optimal Step Model

As a reminder, the Optimal Step Model simulates pedestrian movement by locally optimizing an utility function at each time step. For each pedestrian, we look into a circle around them for the best position to move to, and this search is discretized by evaluating some key points around the pedestrian rather than all the possibilities.

There are two sets of parameters that we can vary: the parameters influencing the value of the utility function, and the parameters influencing the search of the best position.

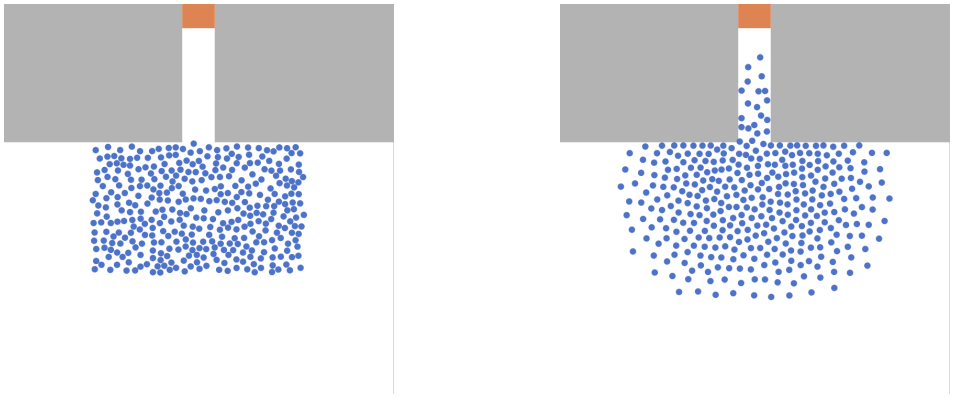
Parameters influencing the utility function The utility function for a pedestrian i takes different values depending on how close the obstacles and the other pedestrians are to this pedestrian i. It depends on the value of the intimate space width s_i and the personal space width s_p. For the obstacle potential, we could also vary the scope of the obstacle repulsion R_o.

Parameters influencing the discrete search When searching for the best next position, we draw a circle of radius r around the pedestrian, corresponding to the step length of this pedestrian. We have seen that this radius is related to the current speed v, see Equation 2.9. We could vary the values of the coefficients in this equation, but they were found based on an empirical study so they are not the most interesting parameters to vary. As seen in Figure 2.5, we then take a finite number of points on and in this circle and evaluate them. We then move to the points with the biggest utility. We have the parameter q which represents the number of points we will evaluate by circle, and k the number of circles we are drawing around the pedestrian. We will investigate the influence of these two parameters on the simulations.

5.2.1.1 Parameters of the Utility Function

When running simulations with a dense group of pedestrians, we noticed an unexpected behavior: the simulated pedestrians move away from each other just after the beginning of the simulation, even if they step away from their destination. After that, they are normally attracted by the target. This is illustrated in Figure 5.9.

We assume that this behavior is caused by the personal space of each pedestrian. Indeed, the potential induced by the influence of nearby pedestrians is significant in the personal zone, and is stronger than the attraction to the target. This is not



(a) Before simulation starts **(b) Just after the simulation starts**

Figure 5.9: Explanation of the unexpected behavior

very realistic, especially in case of an emergency where people tend to rush towards the exits.

We can vary two parameters to try to solve this issue: the `pedPotentialIntimateSpaceWidth` corresponding to the intimate space of pedestrians (s_i), with default value of 0.45m, and the `pedPotentialPersonalSpaceWidth` corresponding to the personal space of pedestrians (s_p), set to 1.2m. We tested three values for the intimate space: 0m, 0.2m and 0.45m and three values for the personal space: 0m, 0.6m and 1.2m. A snapshot of each simulation after 6 seconds of simulation can be found in Appendix C. Figure 5.10 shows the snapshot of the simulations for the intimate space set to 0.2m.

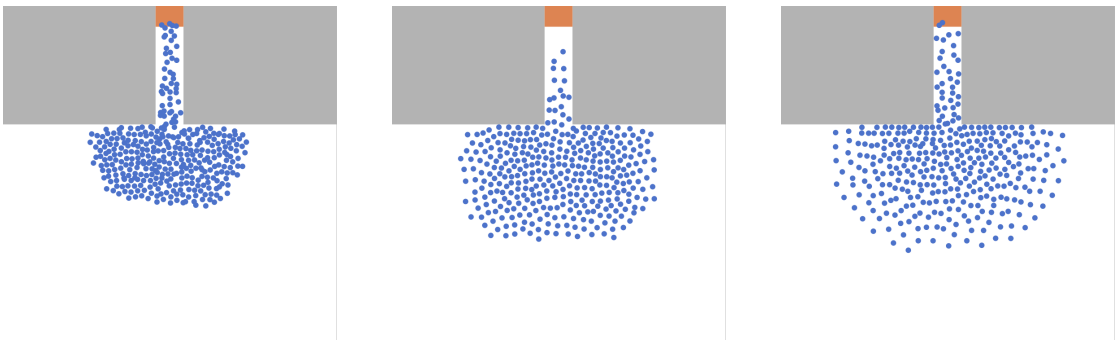


Figure 5.10: Intimate space: 0.2m; Personal space: 0m(left), 0.6m(middle), 1.2m(right)

We noticed that when decreasing both parameters, the simulated pedestrians move less away from each other which is what we expected. We also see that the personal space has a more significant influence than the intimate space on the

results. We can observe that reducing the personal space of pedestrians creates more compact crowds during the simulation. Depending on how urgent is the situation in real life, people will let more or less space between each other. If it is a big emergency with for example an explosion, people will probably push each other to try to go out fastest, and the scenario that comes closest to reality will be the one on the left of Figure 5.10. On the other hand, in a less urgent situation the real-life scenario could look like the one in the middle, or even the one on the right. We can say that these parameters will have to be set according to the level of emergency of the situation we are simulating.

We wanted to investigate whether this parameter variation influences other metrics, as the number of collisions or the evacuation time of pedestrians. As changing these parameters results in a more or less dense crowd during simulation, it would be logical if the number of collisions varies too. And for the evacuation time, we can expect that the most compact crowd evacuates the fastest, but this may not always be the case. We ran a simulation of 350 pedestrians moving toward one exit in a bottleneck, and we collected the number of collisions of our nine simulations in Table 5.5.

		s_i [m]		
		0	**0.2**	**0.45**
s_p [m]	**0**	22195	18255	13770
	0.6	5049	2319	182
	1.2	3657	1653	300

Table 5.5: Number of collisions

Once again, we see that the personal space has a bigger influence that the intimate space. And as predicted, when we reduce the spaces of pedestrians, there are more collisions.

We also took a look at the total time of the evacuation of all pedestrians, for the same scenario as for the collisions, and reported it in Table 5.6. We can observe

		s_i [m]		
		0	**0.2**	**0.45**
s_p [m]	**0**	98.8	98	100.4
	0.6	132	135	175.2
	1.2	108	110.4	133.2

Table 5.6: Evacuation time in seconds

from these data that the intimate space only slightly influences the evacuation time, and we see a general increase of the evacuation time with the radius of the intimate

space. On the other hand, we cannot deduce a tendency in the time variation when changing the width of the personal space. To have a better understanding, we collected the evacuation time of more simulations varying the personal space from 0m to 1.2m, increasing it every time of 10 cm. We set the intimate space radius to 0.45m. We gathered the results in a graph shown in Figure 5.11.

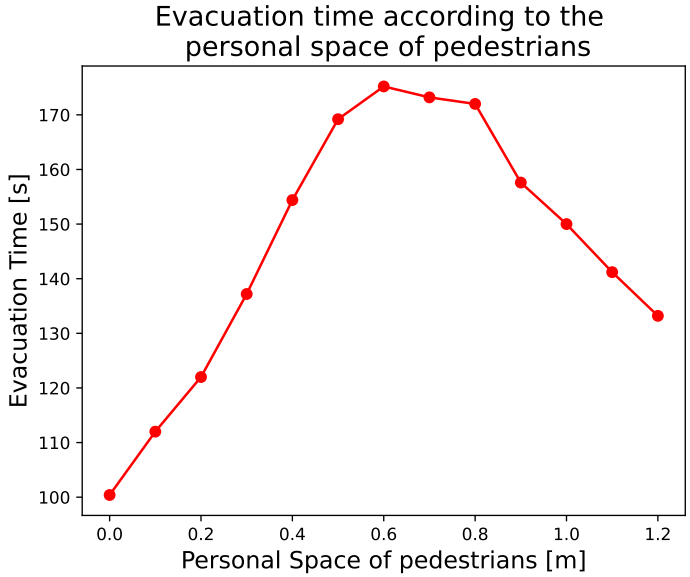
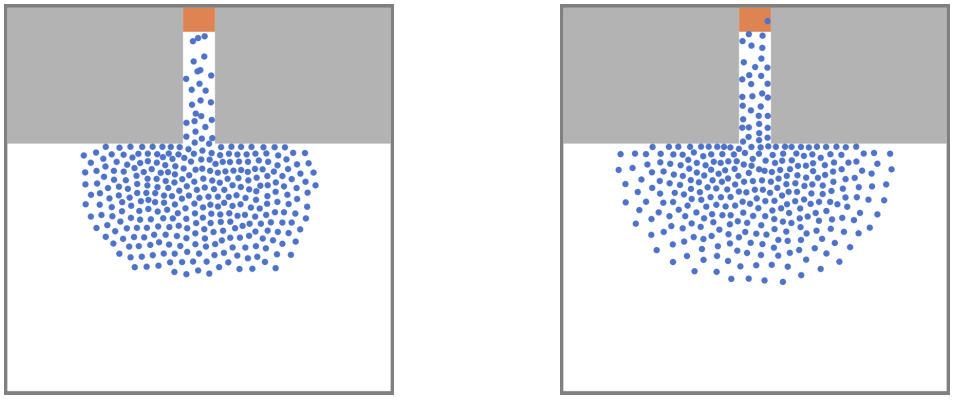


Figure 5.11: Evacuation time according to the personal space radius

We observe that the evacuation time does not change in a linear way. When the personal space radius increases from 0 to about 0.7 meters, the evacuation becomes slower. This happens because as the distance between pedestrians increases, they cannot stand very close to each other anymore, so fewer pedestrians can reach the exit at the same time. The group becomes more and more spread out, but not in an efficient way, and the movement slows down.

However, after 0.7 meters, when the distance continues to increase, the evacuation time starts to go down. This is because pedestrians begin to spread out in a more regular and organized shape, often forming a kind of semicircle in front of the bottleneck. That phenomenon pushed pedestrians even move closer to the side walls. This better organization allows people to leave more smoothly, without blocking each other, even though the group is less dense. This behavior is represented in Figure 5.12. We took a snapshot after 20 seconds of simulation. The left image corresponds to a personal space radius of 0.7m, and the one on the right corresponds to a radius of 1.2m.

In conclusion, the evacuation is the slowest around 0.7 meters, when the space between pedestrians is enough to reduce the crowd density but not yet enough to



(a) s_p = **0.7m** (b) s_p = **1.2m**

Figure 5.12: State of the simulation after 20 seconds.

improve the movement. After that, the larger distance helps the group flow better, and the evacuation becomes faster again.

5.2.1.2 Parameters of the Local Search

The next parameters to vary are the ones used in the local search of the next best position. By default in Vadere, pedestrians evaluate 4 points on one circle of radius r, the step length of pedestrians. But we have seen in Section 2.1.2 that we could evaluate points on several circles within the circle of radius r, and we could evaluate more than 4 points by circle. If we increase the number of points evaluated, logically the execution time should increase, but pedestrians could find their way to their destination faster, and that could influence the evacuation time.

We tried the same bottleneck scenario as for the variation parameter of the utility function with the default parameters of intimate and personal space. We tried with a growing number of circles: 1, 2 and 3; and a growing number of evaluation points on each circle: 4, 8, 16 and 32. We collected the evacuation time (Table 5.7), the total number of iterations of the simulation (Table 5.8), and the execution time of each simulation (Table 5.9).

		# circles		
		1	**2**	**3**
# points	**4**	135	130.8	132.4
	8	127.6	126.8	127.2
	16	124.4	124.8	123.6
	32	123.6	122.4	124.8

Table 5.7: Evacuation time in seconds

We can observe from the data in Table 5.7 that the number of circles does not influence much the evacuation time. On the other hand, when we evaluate more points on each circle, we see that the evacuation time slightly decreases. This means that evaluating more positions allows pedestrians to find their way to the target faster. This is related to the number of iterations.

		# circles		
		1	**2**	**3**
# points	**4**	340	329	333
	8	321	319	320
	16	313	314	311
	32	311	308	314

Table 5.8: Number of iterations for each simulation

The number of iterations decreases when the number of evaluation points increases, which shows again that pedestrians reach their target faster when they can choose between more different positions. We clearly see that increasing the number of evaluation points improve the efficiency of the evacuation, but at which cost? Does the execution time significantly increases? Table 5.9 shows the different execution times for each simulation.

		# circles		
		1	**2**	**3**
# points	**4**	11.3	10.117	10.473
	8	11.23	11.81	13.952
	16	15.148	17.33	21.178
	32	25.422	29.167	31.929

Table 5.9: Execution time in seconds

As expected, the execution time grows with the number of points to consider. When evaluating 32 points on a circle, the increase of the evacuation time is significant, almost multiplied by a factor 3, compared to the improvement of the evacuation time, which is only a few seconds. We could find a compromise and take 8 or 16 points by circle, because the execution time stays reasonable and the evacuation time is still better than the default values.

5.2.2 Gradient Navigation Model

The Gradient Navigation Model uses gradients of distance functions to guide pedestrians toward their destination. There are less parameters that we can vary in

this model. For the obstacle and pedestrians influence, the gradients are computed thanks to a smooth function, see Equation 2.27. In this function, we have to define a scaling factor p and a support R. The scaling factor influences the value of the potential, the greater p is, the stronger will the potential be. The support is the radius of the zone where the repulsion is not null, and we will vary it to see its influence on the simulations.

As mentioned in the mathematical description of the model, the obstacle repulsion must not be too strong otherwise pedestrians would not be able to walk through corridors. And to compensate this weak repulsion, the area of repulsion is bigger. We will then not vary the parameters related to the obstacle potential.

The scaling factor for the pedestrian floor field p_p has a default value of 2.72, and we observe in Table 5.10 the reaction of the simulation to the variation of this parameter.

p_p	**Evacuation time [s]**	**Number of collisions**
1	70.4	42116
2	97.2	22166
3	105.2	4137
4	114	344
5	122.4	110
6	131.6	151

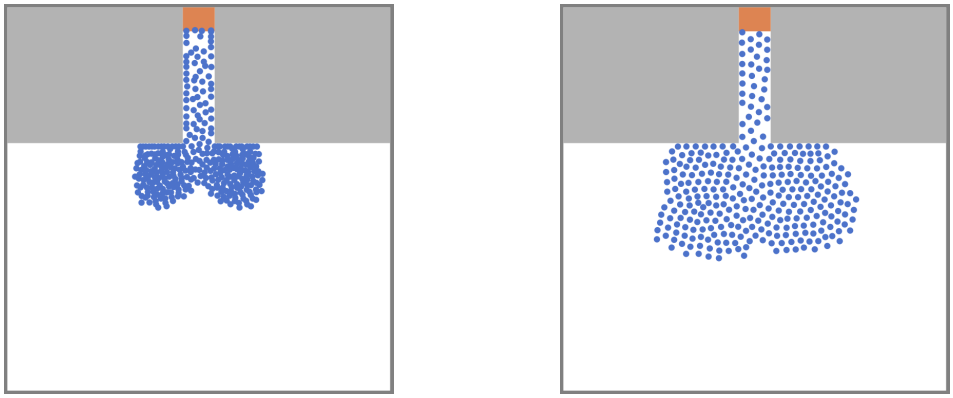
Table 5.10: Variation of parameter p_p

We see that the evacuation time increases with the scaling factor. We could explain that by the fact that when the repulsion is weak, the attraction to the target is stronger than the repulsion of other pedestrians, and the they push each other towards the target. We noticed that when the scaling factor is too small, the simulation becomes unrealistic, because pedestrians overlap too much as shown in Figure 5.13. We can also see this behavior with the number of collisions in Table 5.10.

We can also vary the support R_p, which corresponds to the radius of repulsion of pedestrians. The default value of Vadere is set to 0.8m. We tried the same scenario with different values of support: 0.2m, 0.5m, 0.8m and 1.2m. The data are gathered in Table 5.11.

We encounter the same problem as for the scaling factor, when the support is too small, the crowd behavior becomes unrealistic, with too many collisions.

Once again, we can set this parameter according to the real-life scenario we are simulating.



(a) Scaling factor of 1 (b) Scaling factor of 4

Figure 5.13: Snapshots of the simulation with the Gradient Navigation Model

R_p [m]	Evacuation time [s]	Number of collisions
0.2	46.8	110318
0.5	73.6	55438
0.8	104.8	8000
1.2	177.2	3573

Table 5.11: Variation of parameter R_p

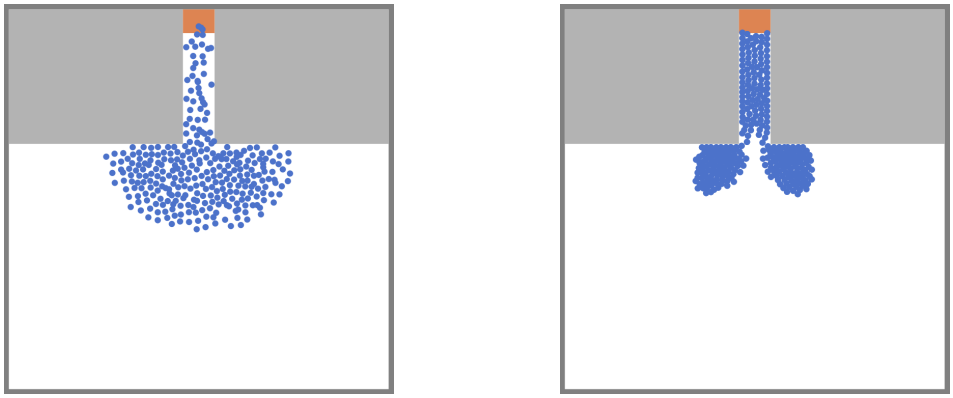
5.2.3 Discussion

We found several interesting results in this parameter variation. First of all, choosing the minimal distance to keep between pedestrians has an important impact on the evacuation, and on how the crowd organizes itself, and that for the two observed models. When the distance is small, the density becomes very high for both models. For the Optimal Step Model, the crowd stays "ordered", unlike for the Gradient Navigation Model, where pedestrians tend to gather at the edges of the bottleneck entrance instead of staying as a uniform crowd, as showed in Figure 5.14.

The distance parameters should be set according to the emergency level of the situation in order to match the simulation the best with the reality.

For the Gradient Navigation Model, we also investigated the influence of the value of the repulsion on the simulations, and we concluded that a stronger repulsion keeps pedestrian away from each other, which is logical, but this also influences directly the evacuation time.

Finally, for the Optimal Step Model, the number of points pedestrians consider to find their next position is very important, as it allows them to find their way to the target more or less rapidly. It is important to find a balance between the quality of the evacuation and the computational efficiency.



(a) Optimal Step Model **(b) Gradient Navigation Model**

Figure 5.14: Simulations with small inter-pedestrians distance

In conclusion, we can say that both models could be suitable for modeling the evacuation of large places as for the *24h vélo of Louvain-la-Neuve*, but the Optimal Step Model presents a few more advantages than the Gradient Navigation Model. First, it is more flexible than the GNM, because we can vary more different parameters in different parts of the model. Then, it is faster for big-scale scenario with a complex structure and several obstacles. Indeed, we have seen that adding obstacles in the topography increases more the execution time for the GNM than for the OSM, and the scenario we used in the scalability study is rather simple compared to real-life scenarios. The Gradient Navigation Model could become really slow in simulations with a lot of obstacles. Furthermore, for some configuration of parameters, the GNM offers some unrealistic crowd behavior. This is why we would choose the Optimal Step Model for our simulations, even if the Gradient Navigation Model is also a good choice.

5.3 Comparison with real data

The best way to validate our chosen model is to confront it to real data, such as data collected in well conducted experiments. A lot of studies over pedestrian dynamics have been carried out: to prove some specific behaviors in crowds ([Gar+14], [App+18]) or even to investigate the influence of the geometry of bottlenecks on the pedestrian flow ([RKS11], [Sey+09], [Lid+11], [KGS06], [Lia+14]). The main inconvenient of these studies is that the raw data are not available. Indeed, these articles studied the movement of pedestrians in laboratory conditions, and searchers often filmed the group of pedestrians and/or collected some data such as the trajectories of pedestrians, their velocities, etc. , but these data are not accessible. We have to find another way to compare our model with empirical data.

As the effect of the bottleneck width on the pedestrian flow has widely been studied, and an important part of the evacuation simulations is the crowd behavior in front of emergency exits, we could reuse the results of these studies to see if our model produces similar results. This idea is taken from the article [Lia+14], that compares different results of different studies. The comparison is done based on how the bottleneck width influences the flow of pedestrians.

We tried to use the same experimental setup as for [Lia+14], which is 350 pedestrians standing just in front of a bottleneck at the start of the simulation. The bottleneck length is 1 m and we decided to vary the width between 1 m and 4 m, increasing it each time by 50 cm. To compute the flow, we took the time difference between the first and the last pedestrian that crosses the line representing the entry of the bottleneck, and divided the number of pedestrians (350) by the time difference. We compared our results with the results of four different studies: [Sey+09], [RKS11], [KGS06] and [Lia+14], and reported it on a graph shown in Figure 5.15.

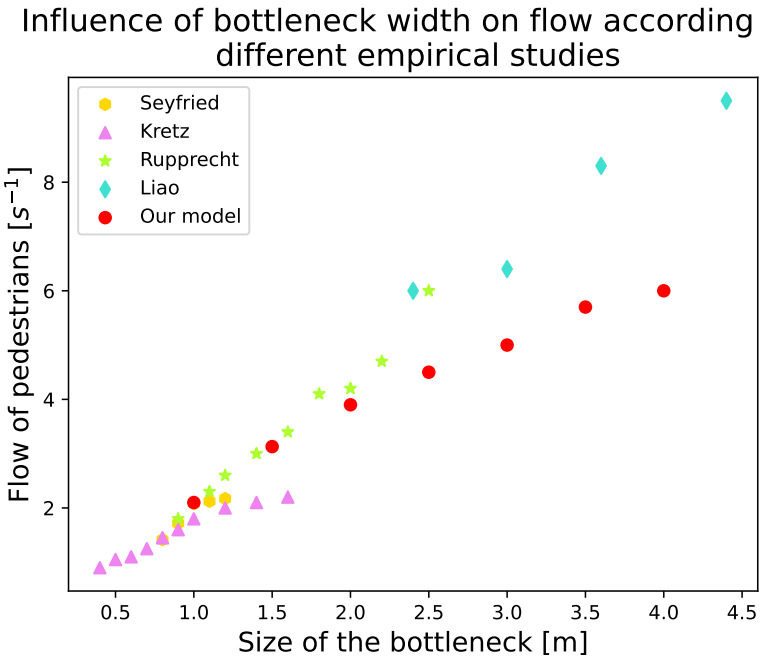


Figure 5.15: Influence of the bottleneck width on the pedestrian flow according to different studies, and according to our model

Note that for Rupprecht ([RKS11]), Kretz ([KGS06]) and Liao ([Lia+14]), the values were read from graphs and do not come from numerical data, which adds some inaccuracies. The experimental setup differs from one study to another, the main difference is the number of pedestrians (120 for Rupprecht, 94 for Kretz and

40 for Seyfried) but this is taken into account in the computation of the flow. The bottleneck length also differs, but [Lia+14] shows that this does not influence the pedestrian flow.

We can observe from Figure 5.15 that the relation between the bottleneck width and the pedestrian flow is linear. Even if for larger bottleneck width, the simulated flow seems to be a little lower than for Liao ([Lia+14]), we can see that our model follows the tendency of the empirical studies. Note that all found studies simulate pedestrians in laboratory conditions, and then do not include panic effects caused by an emergency situation.

Conclusion

6

This thesis is focused on the development of a pedestrian evacuation simulation tool, specifically designed for the *24h vélo of Louvain-la-Neuve*. This event gathers thousands of students in a small urban space, and managing the movement of crowds, especially in emergency situations, is a challenge. The goal of this work was to design a reliable simulation system that could help understand pedestrian behavior, test different scenarios, and improve safety planning through virtual simulations.

In the first part of the project, we studied four well-known microscopic pedestrian models: the Optimal Step Model (OSM), Cellular Automaton (CA), Social Force Model (SFM), and Gradient Navigation Model (GNM). Each model was described and analyzed mathematically, from the assumptions they make to their equations of motion. We found out that each model has its own characteristics. The OSM guide pedestrians towards their target by optimizing each step, the CA is simple and fast but lacks of precision, the SFM uses social forces to simulate interactions but becomes very computationally expensive, and the GNM allows pedestrians to follow a potential field based on navigation goals.

In the second chapter, we described a set of metrics that will be used to compare the models in a consistent way. These metrics are based on the realism of the simulation, the stability and the computational efficiency. This step is important for identifying the models that are both practical and reliable for simulating evacuations.

To implement and test the models, we looked up some open-source frameworks for pedestrian simulations and selected Vadere as the foundation for our tool. It supports several pedestrian models, including those we studied, and offers a graphical user interface and flexible scenario configuration. We explored the internal structure of Vadere and documented its components. We also made some little modifications to the software to adapts it to our experimental needs.

In the experimental chapter, we performed multiple simulations using all four models. Based on our metrics and observations, we excluded the Cellular Automaton model due to its lack of realism and the Social Force Model because it required too much computing time and became impractical for large-scale simulations. The final analysis focused on the two remaining models, the Optimal Step Model and the Gradient Navigation Model. We performed parameter variations to better

understand their behavior and adaptability in different evacuation scenarios. The results showed that both OSM and GNM can offer realistic and efficient simulations under certain conditions. OSM performed well in structured environments and was more flexible in terms of variation of parameters, while GNM predicted smoother trajectories. However, the OSM presents some advantages over the GNM. Indeed, it is faster in terms of execution for large-scale scenarios containing obstacles. It is also more flexible in terms of variation of parameters, and the GNM simulates unexpected crowd behaviors for a certain choice of parameters. For all these reasons, we decided to keep the Optimal Step Model.

The last step of this work was to compare the chosen model with some empirical data found in several studies. We investigated the influence of the width bottleneck on the crowd behavior. Even if the experimental conditions did not match exactly the conditions of our simulations, our simulated model followed the same tendency of the results found in these studies.

Despite these achievements, this study has some limitations. First, the pedestrian behavior was modeled with simplifying assumptions: we did not include emotional reactions, panic, or group behavior. Second, while Vadere is powerful, its has its limits in terms of visual rendering and does not support 3D environments or real-time decision making. And finally, our simulations were not yet validated against real data from past *24h vélo* events.

Future work could include several improvements. For example, we could collect empirical data during a future edition of the event, using video analysis or sensors, to calibrate and validate the models more precisely. More realistic behaviors could be added, such as panic, group dynamics, or interactions with emergency services. On the software side, developing a 3D or virtual reality visualization system would improve the communication of results to event planners and decision-makers. Lastly, the simulation tool could be extended to similar use cases like concerts, train stations, or sports stadiums.

In conclusion, this thesis presents a complete and flexible framework for simulating pedestrian evacuation during a large public event. By combining theoretical analysis, software adaptation, and simulation experiments, we have shown that it is possible to model evacuation scenarios in a realistic and useful way. The tool developed here represents a solid foundation that can support safety planning and risk management in future editions of the *24h vélo* and other crowded events.

Bibliography

[App+18] C. Appert-Rolland, J. Pettré, A.-H. Olivier, W. Warren, A. Duigou-Majumdar, E. Pinsard, and A. Nicolas. "Experimental study of collective pedestrian dynamics". In: *arXiv preprint arXiv:1809.06817* (2018).

[CBM16] S. Curtis, A. Best, and D. Manocha. "Menge: A Modular Framework for Simulating Crowd Movement". In: *Collective Dynamics* 1 (Mar. 2016), pp. 1–40. DOI: 10.17815/CD.2016.1.

[CSC09] U. Chattaraj, A. Seyfried, and P. Chakroborty. "Comparison of pedestrian fundamental diagram across cultures". In: *Advances in complex systems* 12.03 (2009), pp. 393–405.

[DDH13] D. C. Duives, W. Daamen, and S. P. Hoogendoorn. "State-of-the-art crowd motion simulation models". In: *Transportation research part C: emerging technologies* 37 (2013), pp. 193–209.

[DK14] F. Dietrich and G. Köster. "Gradient navigation model for pedestrian dynamics". In: *Phys. Rev. E* 89 (6 June 2014), p. 062801. DOI: 10.1103/PhysRevE.89.062801.

[Dul+11] L. A. Dultz, S. Frangos, G. Foltin, M. Marr, R. Simon, O. Bholat, D. A. Levine, D. Slaughter-Larkem, S. Jacko, P. Ayoung-Chee, et al. "Alcohol use by pedestrians who are struck by motor vehicles: how drinking influences behaviors, medical management, and outcomes". In: *Journal of Trauma and Acute Care Surgery* 71.5 (2011), pp. 1252–1257.

[Gar+14] A. Garcimartín, I. Zuriguel, J. Pastor, C. Martín-Gómez, and D. Parisi. "Experimental evidence of the "faster is slower" effect". In: *Transportation Research Procedia* 2 (2014), pp. 760–767.

[Gol69] J. W. Goldman. "The Hidden Dimension" by Edward T. Hall (Book Review)". In: *Et Cetera* 26 (1969), p. 241.

[Har10] D. Hartmann. "Adaptive pedestrian dynamics based on geodesics". In: *New Journal of Physics* 12.4 (2010), p. 043032.

[HM95] D. Helbing and P. Molnár. "Social force model for pedestrian dynamics". In: *Phys. Rev. E* 51 (5 May 1995), pp. 4282–4286. DOI: 10.1103/PhysRevE.51.4282.

[KGS06] T. Kretz, A. Grünebohm, and M. Schreckenberg. “Experimental study of pedestrian flow through a bottleneck”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2006.10 (Oct. 2006), P10014. DOI: 10.1088/1742-5468/2006/10/P10014.

[KH17] W. Kang and Y. Han. “A simple and realistic pedestrian model for crowd simulation and application”. In: *arXiv preprint arXiv:1708.03080* (2017).

[KHB13] A. Kneidl, D. Hartmann, and A. Borrmann. “A hybrid multi-scale approach for simulation of pedestrian dynamics”. In: *Transportation research part C: emerging technologies* 37 (2013), pp. 223–237.

[Kle+19] B. Kleinmeier, B. Zönnchen, M. Gödel, and G. Köster. “Vadere: An Open-Source Simulation Framework to Promote Interdisciplinary Understanding”. In: *Collective Dynamics* 4 (Jan. 1, 2019). DOI: 10.17815/CD.2019.21. published.

[Kre19] T. Kretz. “An overview of fundamental diagrams of pedestrian dynamics”. In: *DOI: https://doi. org/10.13140/RG* 2.30070.96326 (2019).

[Lia+14] W. Liao, A. Seyfried, J. Zhang, M. Boltes, X. Zheng, and Y. Zhao. “Experimental Study on Pedestrian Flow through Wide Bottleneck”. In: *Transportation Research Procedia* 2 (2014). The Conference on Pedestrian and Evacuation Dynamics 2014 (PED 2014), 22-24 October 2014, Delft, The Netherlands, pp. 26–33. ISSN: 2352-1465. DOI: https://doi.org/10.1016/j.trpro.2014.09.005.

[Lid+11] J. Liddle, A. Seyfried, B. Steffen, W. Klingsch, T. Rupprecht, A. Winkens, and M. Boltes. *Microscopic insights into pedestrian motion through a bottleneck, resolving spatial and temporal variations*. 2011. arXiv: 1105.1532 [physics.soc-ph].

[Nit13] C. Nitzsche. “Cellular automata modeling for pedestrian dynamics”. In: *Bachelor thesis* (2013).

[Pednd] Pedestrian Dynamics. *Optimal Steps Model*. Consulté le 27 mai 2025. n.d.

[RKS11] T. Rupprecht, W. Klingsch, and A. Seyfried. “Influence of geometry parameters on pedestrian flow through bottleneck”. In: *Pedestrian and Evacuation Dynamics*. Springer, 2011, pp. 71–80.

[Rog+10] C. Rogsch, W. Klingsch, A. Schadschneider, and M. Schreckenberg. *Pedestrian and evacuation dynamics 2008*. Springer, 2010.

[Sey+09] A. Seyfried, O. Passon, B. Steffen, M. Boltes, T. Rupprecht, and W. Klingsch. “New insights into pedestrian flow through bottlenecks”. In: *Transportation Science* 43.3 (2009), pp. 395–406.

[SK12] M. J. Seitz and G. Köster. “Natural discretization of pedestrian movement in continuous space”. In: *Phys. Rev. E* 86 (4 Oct. 2012), p. 046108. DOI: 10.1103/PhysRevE.86.046108.

[SK14] M. J. Seitz and G. Köster. “How update schemes influence crowd simulations”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2014.7 (July 2014), P07002. DOI: 10.1088/1742-5468/2014/07/P07002.

[TH16] E. Treister and E. Haber. “A fast marching algorithm for the factored eikonal equation”. In: *Journal of Computational Physics* 324 (2016), pp. 210–225. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2016.08.012.

[Unind] Université catholique de Louvain. *Les 24h Vélo de Louvain-la-Neuve*. https://www.uclouvain.be/fr/culture/les-24h-velo. Consulté le 22 mai 2025. n.d.

[Vis24] K. Vishwakarma. *Java vs Python vs C++ comparison — coding and performance*. Consulté le 27 mai 2025. Nov. 2024. URL: https://medium.com/@karanvishwakarma/java-python-c-comparison-coding-and-numerical-b25122698930.

[Wag+15] A. K. Wagoum, M. Chraibi, J. Zhang, and G. Lämmel. “JuPedSim: an open framework for simulating and analyzing the dynamics of pedestrians”. In: *3rd Conference of Transportation Research Group of India*. Vol. 12. 2015.

[Wei93] U. Weidmann. “Transporttechnik der fußgänger: transporttechnische eigenschaften des fußgängerverkehrs, literaturauswertung”. In: *IVT Schriftenreihe* 90 (1993).

[Wol83] S. Wolfram. “Statistical mechanics of cellular automata”. In: *Rev. Mod. Phys.* 55 (3 July 1983), pp. 601–644. DOI: 10.1103/RevModPhys.55.601.

[Zha+11] J. Zhang, W. Klingsch, A. Schadschneider, and A. Seyfried. “Transitions in pedestrian fundamental diagrams of straight corridors and T-junctions”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2011.06 (2011), P06004.

[Zha+19] D. Zhang, H. Zhu, S. Hostikka, and S. Qiu. “Pedestrian dynamics in a heterogeneous bidirectional flow: Overtaking behaviour and lane formation”. In: *Physica A: Statistical Mechanics and its Applications* 525 (2019), pp. 72–84. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2019.03.032.

Testing Scenarios
A

A.1 Do pedestrian always reach thier goal?

Figure A.1: Scenario 1

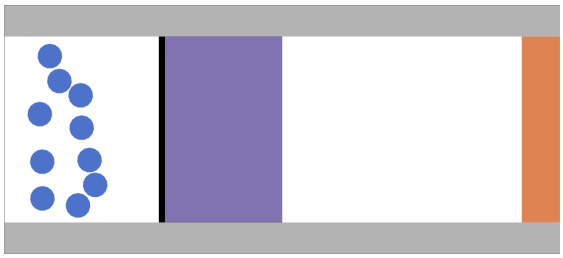


Figure A.2: Scenario 2

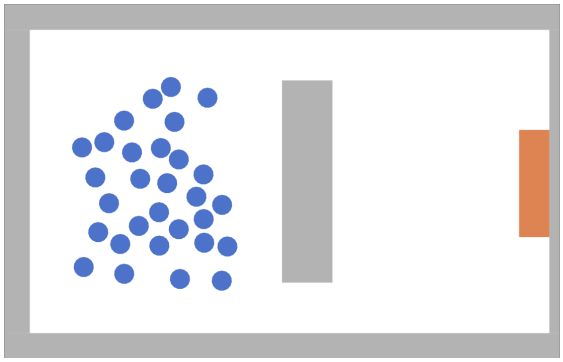


Figure A.3: Scenario 3

Figure A.4: Scenario 4

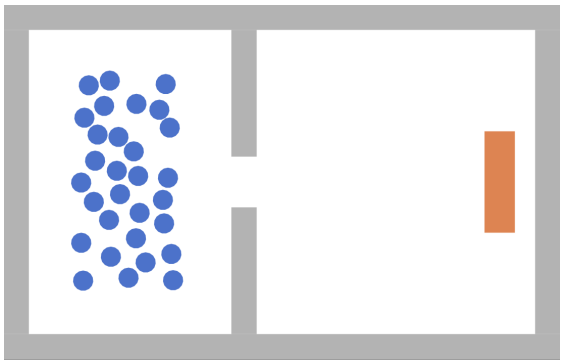


Figure A.5: Scenario 5

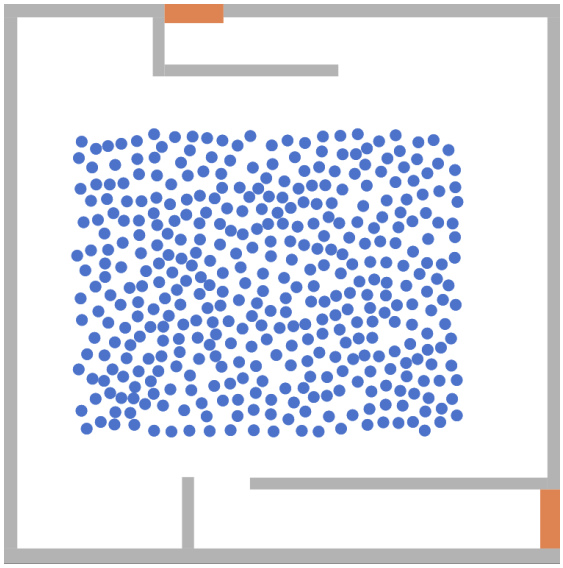


Figure A.6: Scenario 6

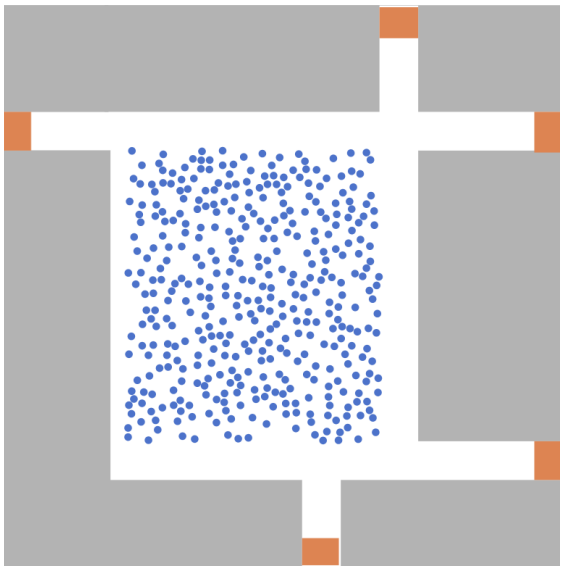


Figure A.7: Scenario 7

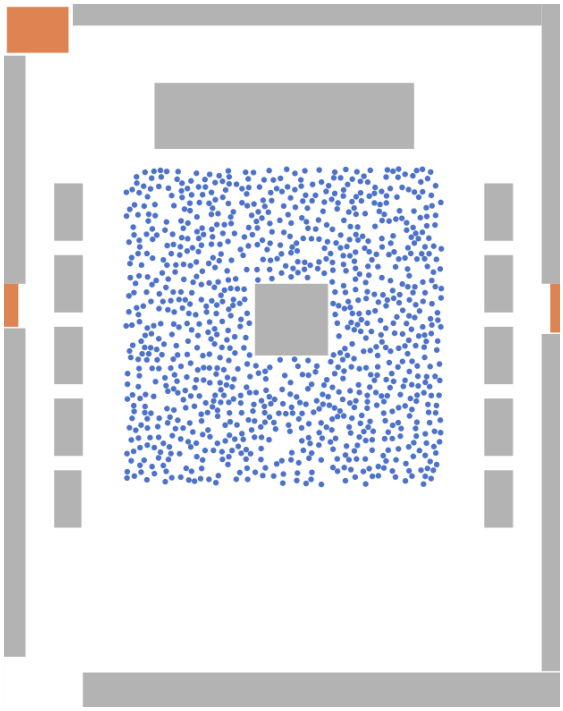


Figure A.8: Scenario 8

A.2 Which model is the most invariant to the random placement of pedestrians

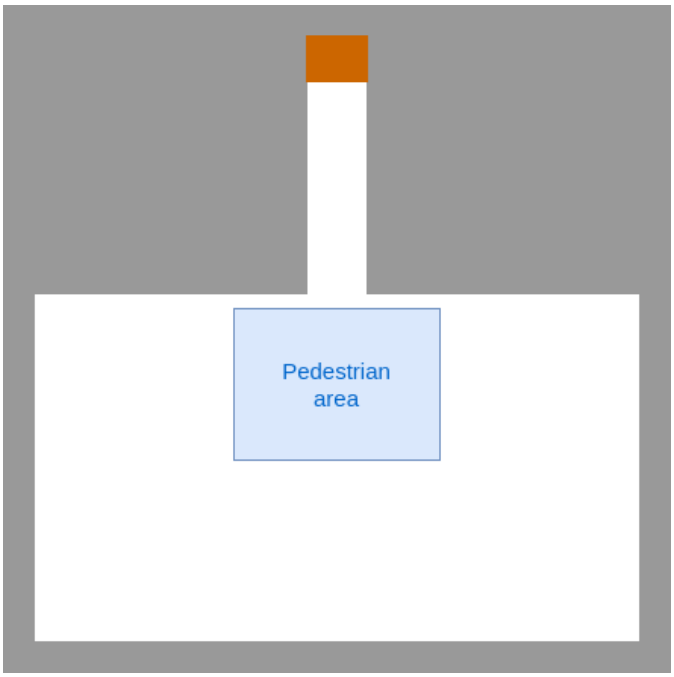


Figure A.9: Scenario for the stability study

A.3 Which model handles the best a growing number of pedestrians?

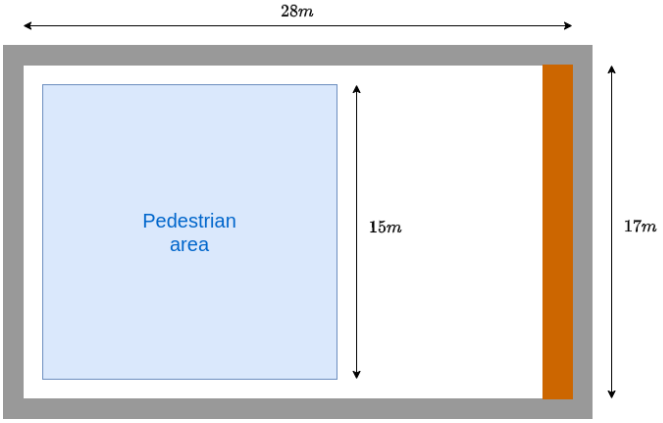


Figure A.10: Simple scenario for the scalability study

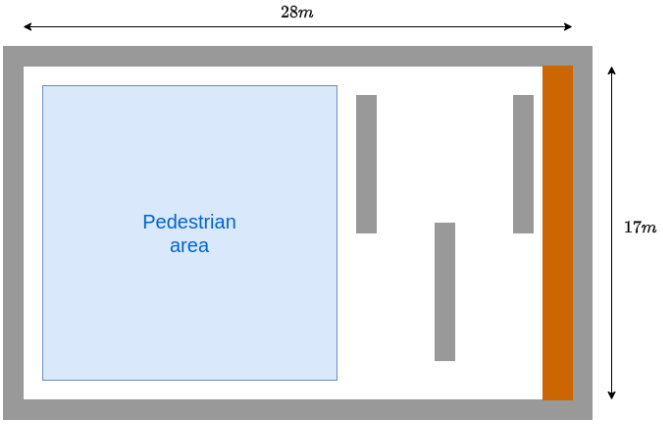


Figure A.11: Complex scenario for the scalability study

A.4 Fundamental Diagram

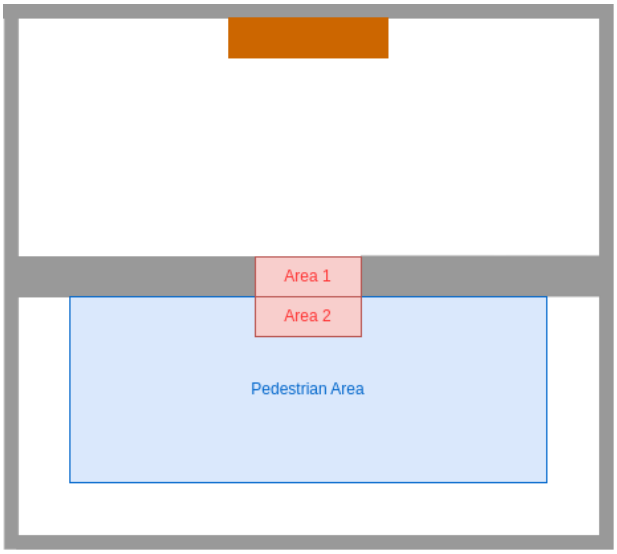
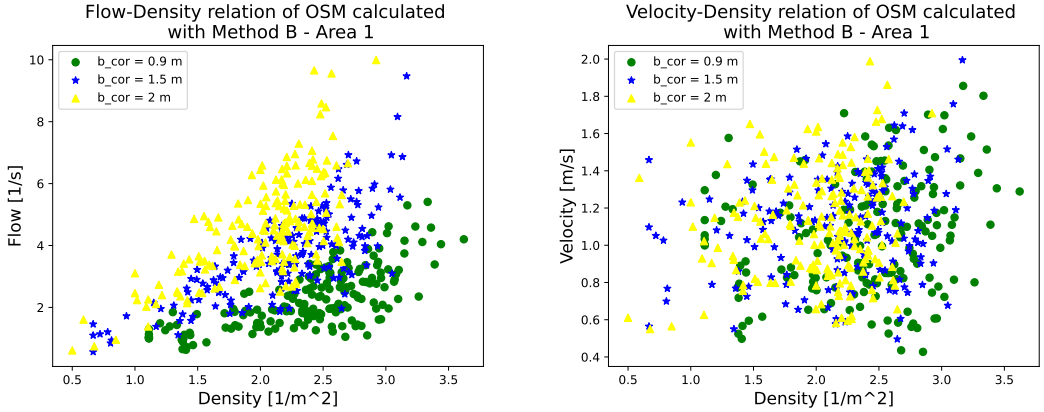


Figure A.12: Topography for the Fundamental Diagram

B Fundamental Diagrams

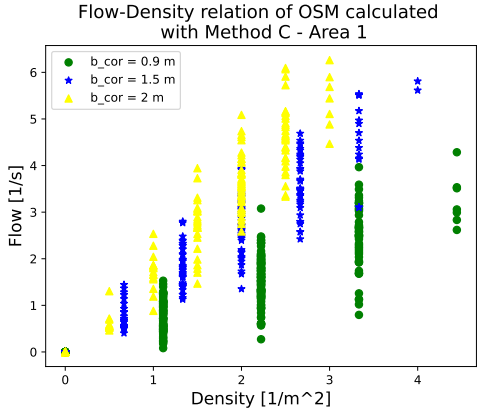
B.1 Optimal Step Model

B.1.1 Area 1

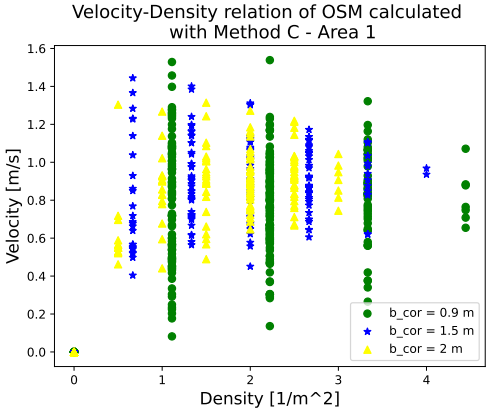


(a) Flow-density relation **(b) Velocity-density relation**

Figure B.1: Method B

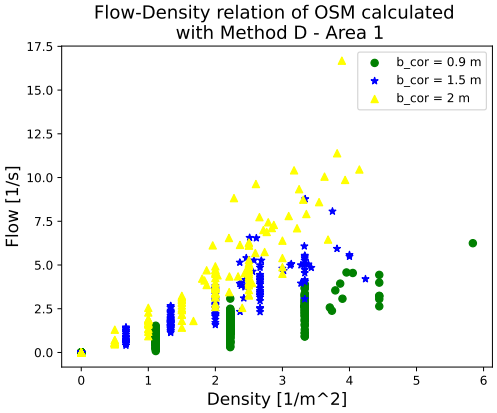


(a) **Flow-density relation**

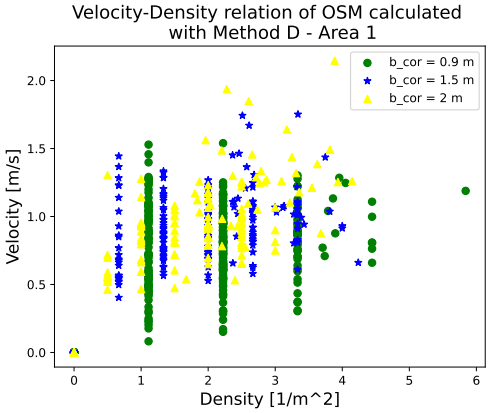


(b) **Velocity-density relation**

Figure B.2: Method C



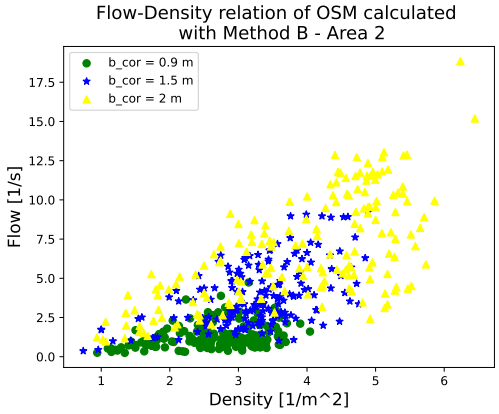
(a) **Flow-density relation**



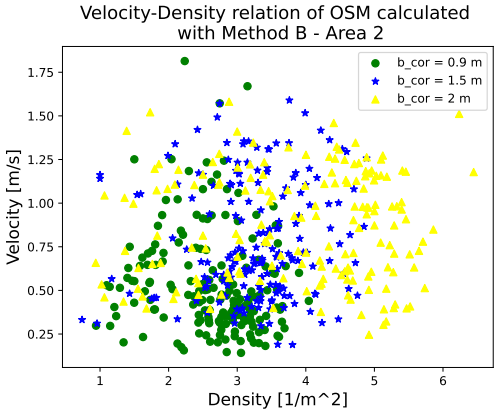
(b) **Velocity-density relation**

Figure B.3: Method D

B.1.2 Area 2

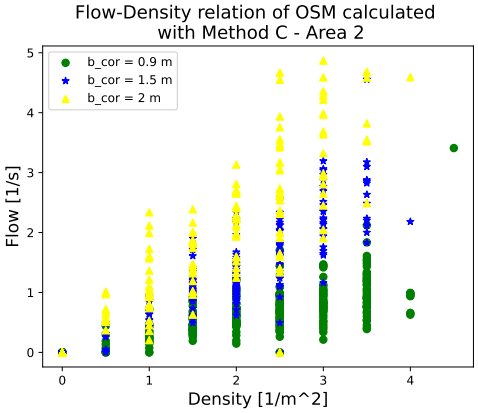


(a) Flow-density relation

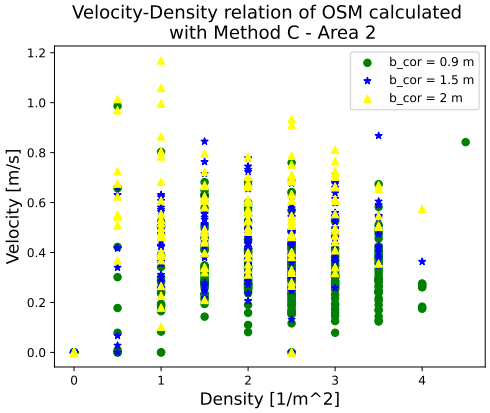


(b) Velocity-density relation

Figure B.4: Method B

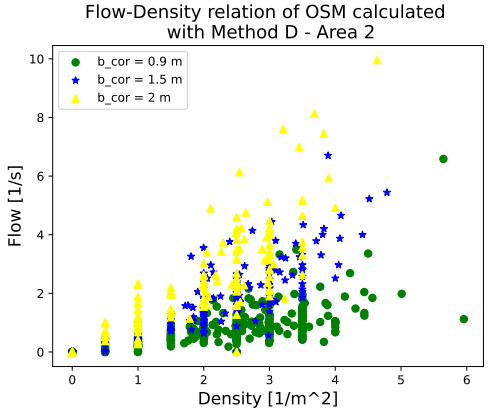


(a) Flow-density relation

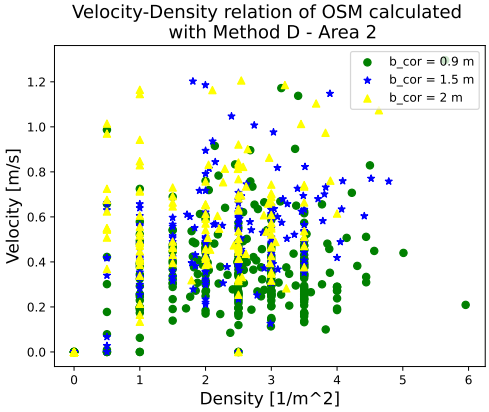


(b) Velocity-density relation

Figure B.5: Method C



(a) **Flow-density relation**

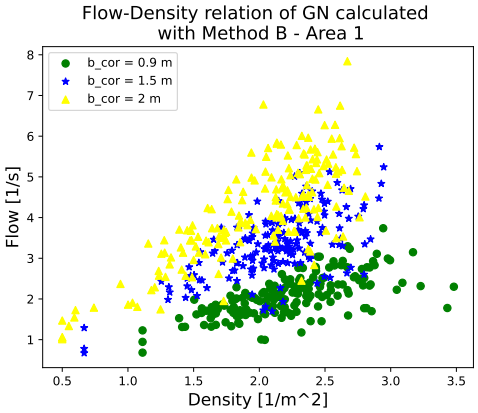


(b) **Velocity-density relation**

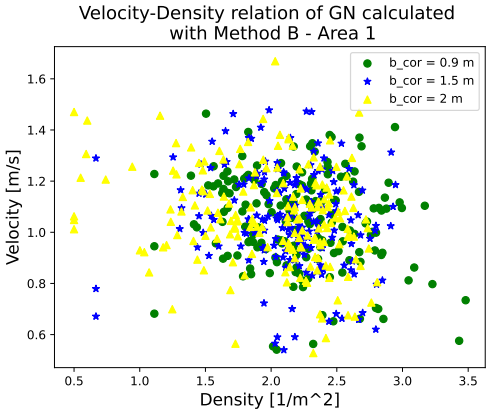
Figure B.6: Method D

B.2 Gradient Navigation

B.2.1 Area 1

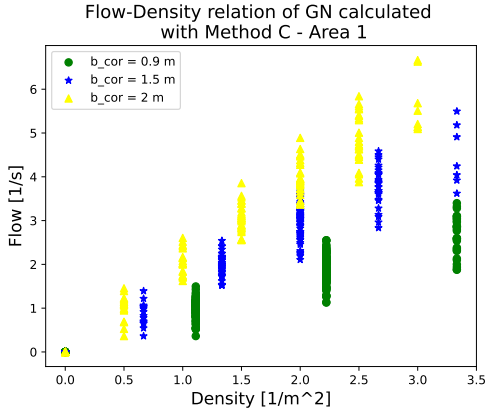


(a) **Flow-density relation**

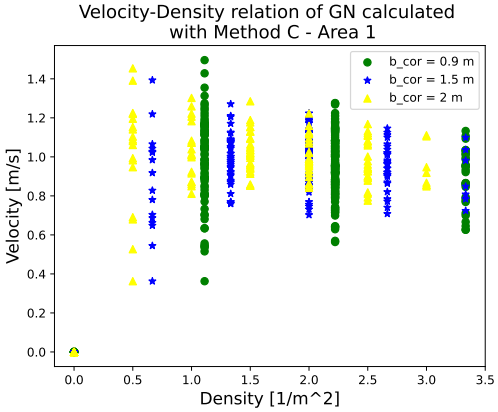


(b) **Velocity-density relation**

Figure B.7: Method B

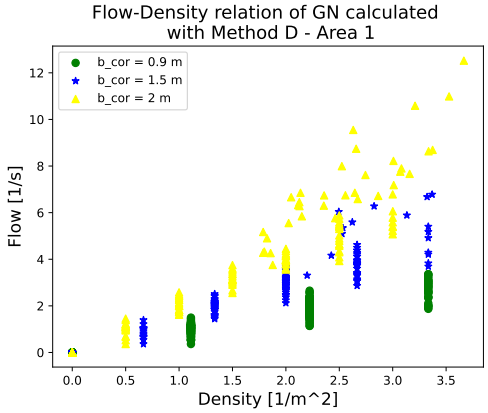


(a) **Flow-density relation**

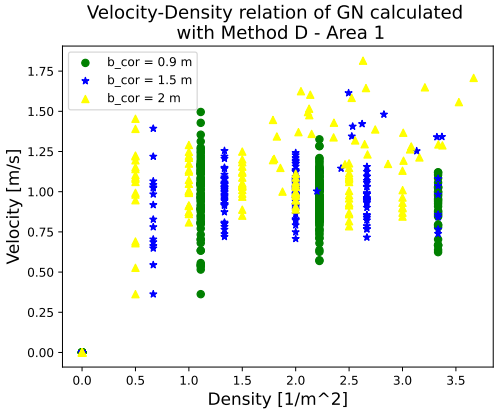


(b) **Velocity-density relation**

Figure B.8: Method C



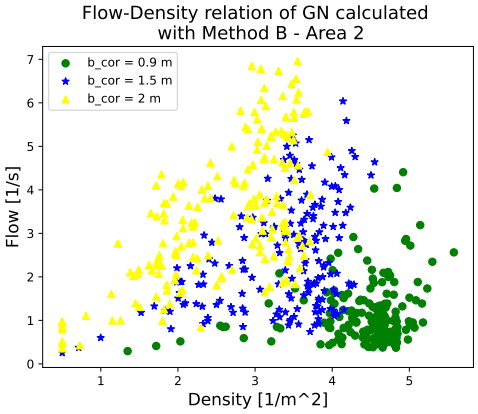
(a) **Flow-density relation**



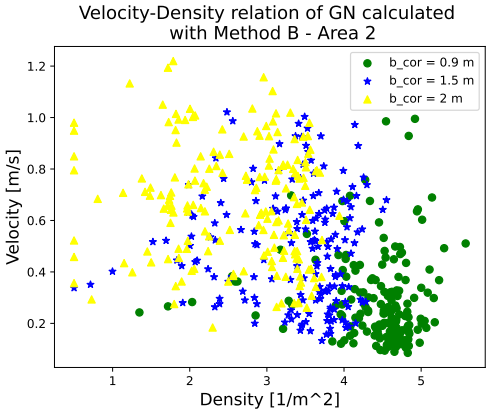
(b) **Velocity-density relation**

Figure B.9: Method D

B.2.2 Area 2

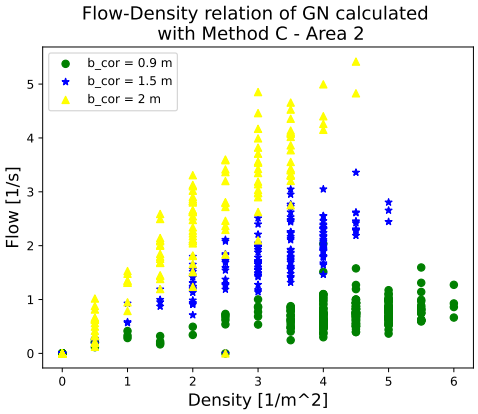


(a) Flow-density relation

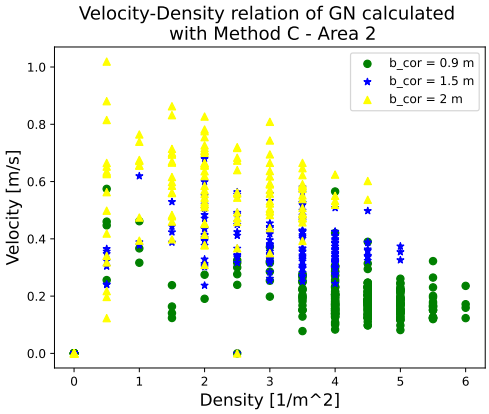


(b) Velocity-density relation

Figure B.10: Method B

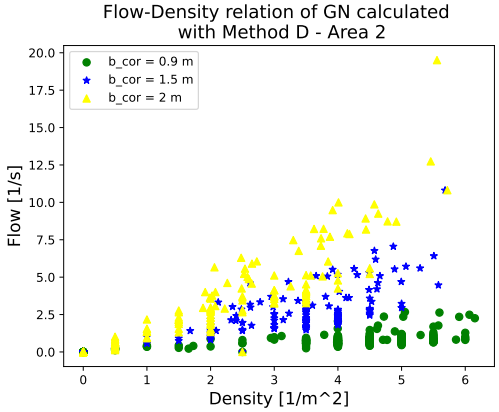


(a) Flow-density relation

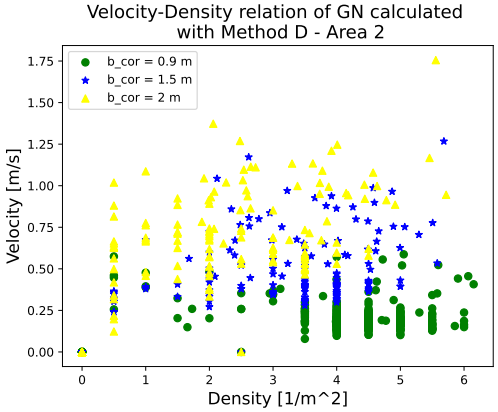


(b) Velocity-density relation

Figure B.11: Method C



(a) **Flow-density relation**

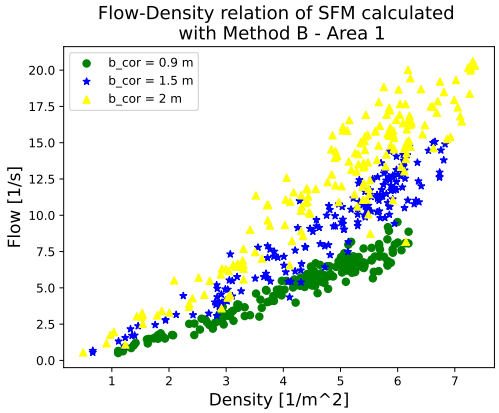


(b) **Velocity-density relation**

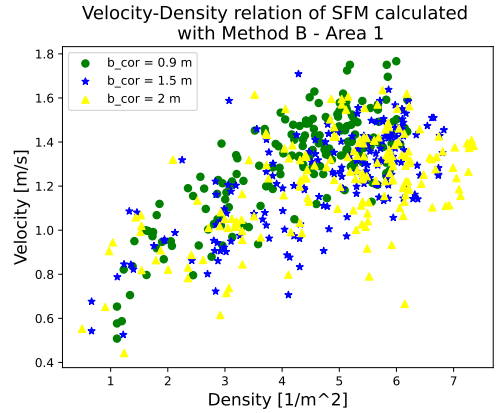
Figure B.12: Method D

B.3 Social Force Model

B.3.1 Area 1

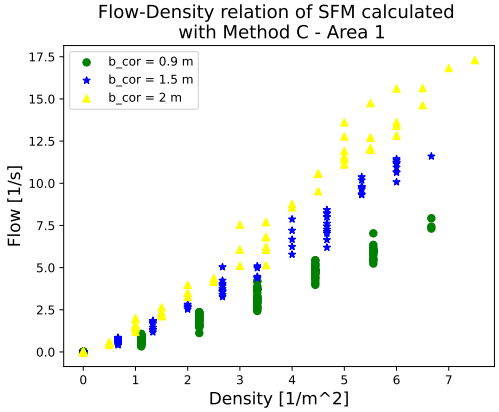


(a) **Flow-density relation**

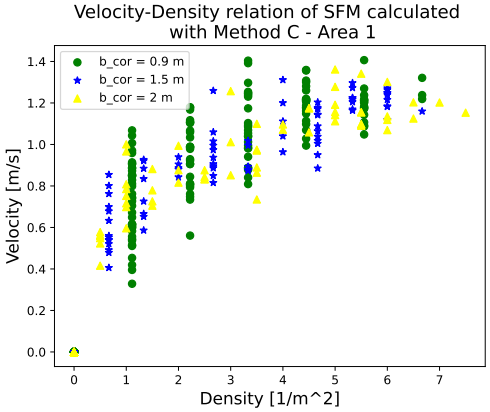


(b) **Velocity-density relation**

Figure B.13: Method B

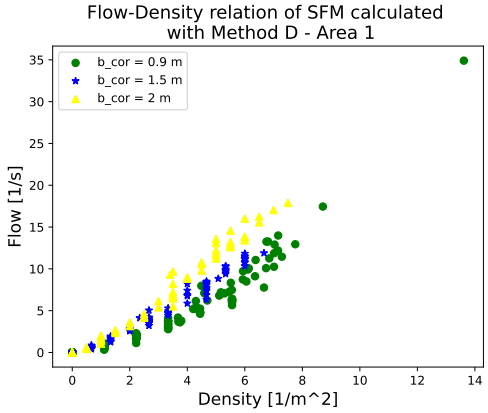


(a) **Flow-density relation**

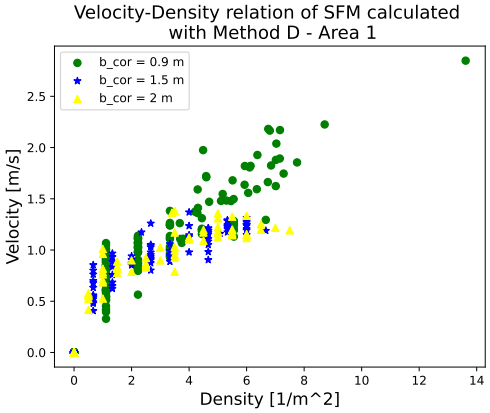


(b) **Velocity-density relation**

Figure B.14: Method C



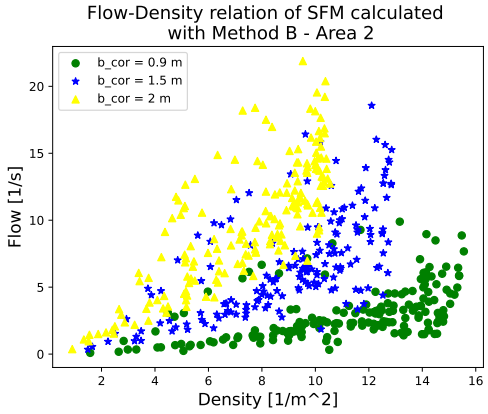
(a) **Flow-density relation**



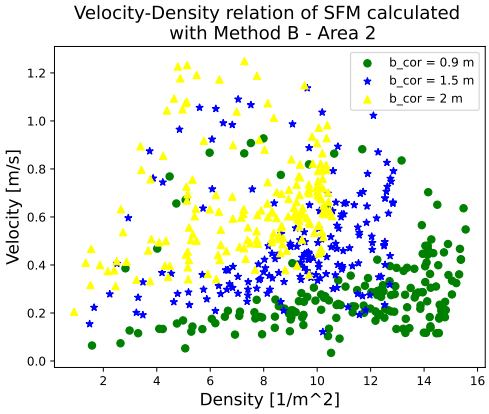
(b) **Velocity-density relation**

Figure B.15: Method D

B.3.2 Area 2

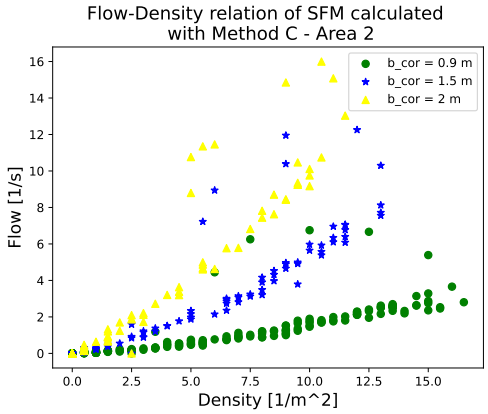


(a) Flow-density relation

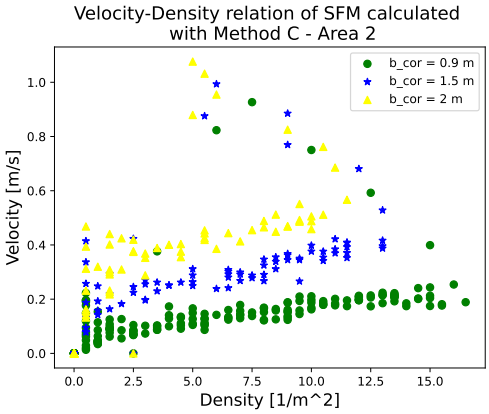


(b) Velocity-density relation

Figure B.16: Method B

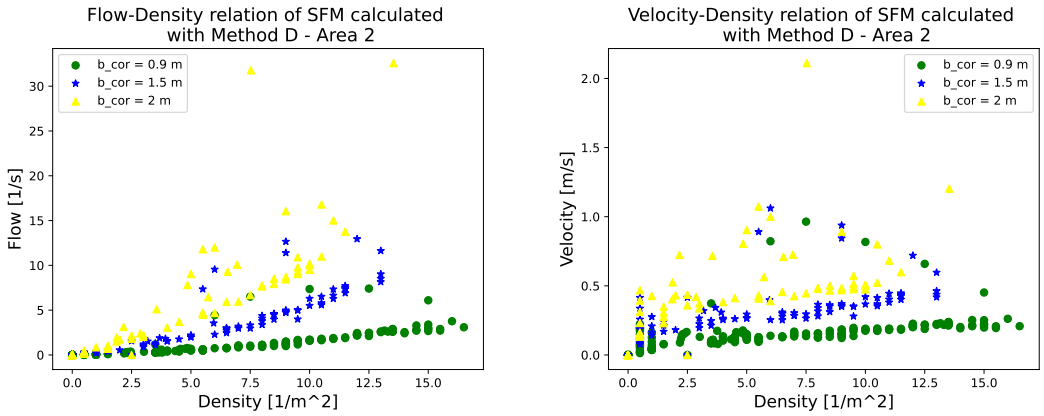


(a) Flow-density relation



(b) Velocity-density relation

Figure B.17: Method C



(a) Flow-density relation **(b) Velocity-density relation**

Figure B.18: Method D

C OSM : Variation of the Intimate and Personal Spaces

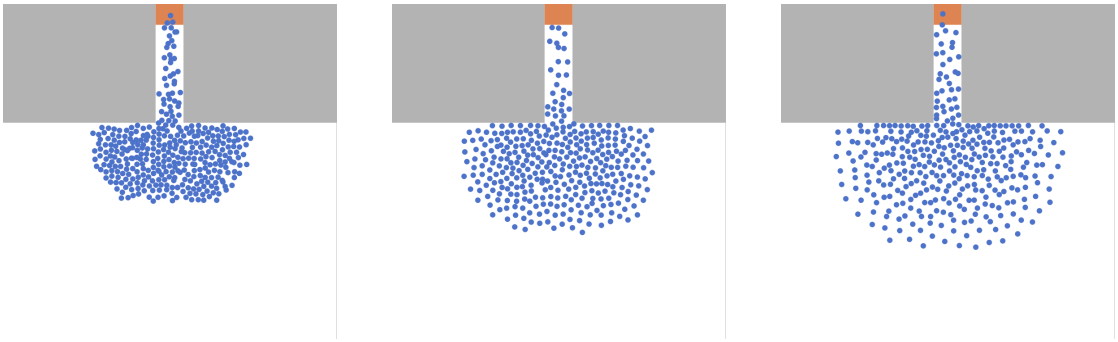


Figure C.1: Intimate space : 0m; Personal space : 0m(left), 0.6m(middle), 1.2m(right)

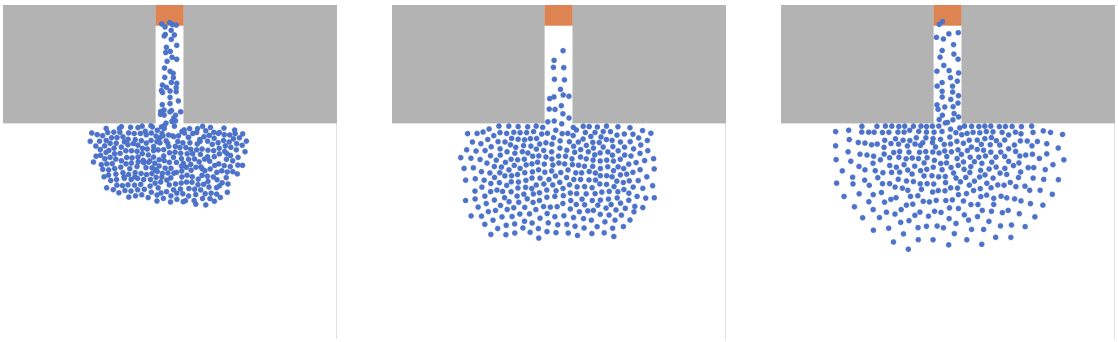


Figure C.2: Intimate space : 0.2m; Personal space : 0m(left), 0.6m(middle), 1.2m(right)

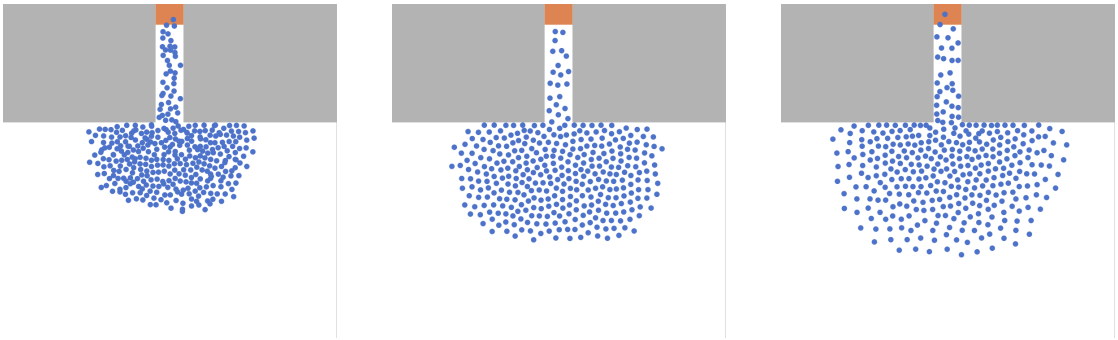


Figure C.3: Intimate space : 0.45m; Personal space : 0m(left), 0.6m(middle), 1.2m(right)

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**