Fachhochschule Aachen

Campus Jülich

Fachbereich: Medizintechnik und Technomathematik,

Studiengang: Technomathematik

Automated Routing in Pedestrian Dynamics

Masterarbeit

vorgelegt von

Arne Graf

Autor:

Arne Graf

Erstprüfer:

Prof. Dr. Johannes Grotendorst

Zweitprüfer:

Dr. Mohcine Chraibi

Jülich, 2015-10-21

Eigenständigkeitserklärung

Diese Arbeit wurde von mir selbstständig angefertigt und verfasst. Es sind keine anderen als die angegebenen Quellen und Hilfsmittel benutzt worden.

Ort und Datum Unterschrift

Diese Arbeit	wurde betreut von:
Erstprüfer:	Professor Dr. Johannes Grotendorst
Zweitprüfer:	Dr. Mohcine Chraibi

Die vorliegende Masterarbeit wurde erstellt am:
Department Civil Security and Traffic,
Jülich Supercomputing Centre,
Forschungszentrum Jülich GmbH.

Abstract

In this thesis, the effect of an alternate floorfield is analyzed, by using it in a newly composed testmodel for pedestrian dynamics. In simulations of pedestrian movement, the routing of agents[1] is an integral part. Routing can be seen as the composition of two aspects: the global pathfinding through a geometry and the avoidance of static or dynamic obstacles (like walls or other agents) in a local situation.

Development of pedestrian simulation shows various models with different answers to the question of navigation. Many of which make use of manually added elements[2] to solve the global pathfinding, which enable the user to simulate crowd movement in that specific geometry. Other models use an algorithm with no restriction to the type of geometry, that will yield a navigation direction. The *Gradient Navigation Model*(GNM) described by Dietrich [7] is one of the later. It uses the solution of the Eikonal equation (see chapter 2.2), which describes a 2-D wave propagation. The wave starts in the target region and propagates throughout the geometry. Agents are directed in the opposite direction of the gradient of a *floorfield*, the aforementioned solution to the Eikonal equation. The Routing using the plain floorfield will yield non-smooth pathways. This could pose a problem for models, relying on a well-posed problem: In [7], Dietrich shows the existance and uniqueness of a solution to his problem-formulation by using the theorem of Picard-Lindelöf. To apply this theorem, Lipschitz-continuous first derivatives of the inputfunctions must exist. This contradicts with non-smooth pathways in a plain floorfield. He solves that problem by introducing a mollifier, which basically takes a locally integrable function and returns a smooth approximation. Thus he creates a well posed problem.

In this thesis, an enhanced floorfield is described, which addresses the aforementioned issue (non-smoothness) as a welcome side-effect. A research group at the Universidad Carlos III de Madrid [14] is working on safe nav-

[1] An agent is the representation of a pedestrian in the simulation.
[2] like a domain decomposition by adding helplines

igation of robots. Since agents should not follow paths, which come close to any obstacle, a distance field is created and used in the Fast Marching algorithm, resulting in *smooth* pathways, which favor a certain distance to walls. The researchers take that approach even further, by transforming any geometry into a skeleton (again using the distance field) and thus having the domain in which the 2-D wave propagates reduced significantly. Their intent is to recalculate the floorfield in real-time using it for the reduced viewfield of a robot's sensors. Our interest in this sleight of hand is different. We take special interest in the behavior of agents close to obstacles. The enhanced floorfield itself yields pathways, which show a wall repulsive character in the negative gradient close to walls. This component is then used in a new model.

It is implemented in JuPedSim[1], a simulation suit for pedestrian simulation, developed at the Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH. It is verified and validated with respect to empirical data. The results seen in the simulations show remarkably good behavior. The model is easy to use, fast and shows an organic routing through complex geometries. The extent to which we alter the floorfield is subject to our analysis.

Contents

1 Pedestrian Dynamics: Introduction

Pedestrian dynamics is a field of research trying to investigate the kinematic and mechanic of pedestrian crowd movement. Understanding, how crowds react in different geometries under various circumstances, enables a safer design of our environment, to best fit the needs of civil and security engineering. Results are applied to safely conduct large events, to create architecture, through which large crowds can safely be moved and to optimize evacuation time in case of an emergency. At the annual hajj[3] in Mecca in 2015, a tragic crowd disaster occurred, where more than 700 pilgrims died and at least 860 more had been injured [2]. Investigation of this incident is not completed. Professor Galea from the Fire Safety Engineering Group at the University of Greenwich predicts, that the frequency of such disasters worldwide will increase due to the higher densities and Urbanization [3].

Pedestrian dynamics provides approaches to plan large events by calculating estimates for capacities of given geometries, researching crowd behavior and applying research results in new designs of civil engineering. To simulate pedestrian crowds, many models exist with different characteristics. Predtetschenskii and Milinkskii [17] are pioneers in pedestrian dynamics, conducting experiments as early as 1969. Few years later, Hirai and Tarui [11] implemented the first known force-based model to simulate crowd behavior. Since then, new models have been described throughout the decades. To maintain orientation, these models can be grouped into classes in the following manner (see figure 1.1):

Macroscopic models tackle crowd behavior without the need to characterize individuals, which make up the crowd. The action of a single agent is neglected and it is assumed, that aggregated values are sufficient to describe the crowd behavior. Metrics, e.g. density or flow, are used to describe the dynamic within the system. Thus a crowd is seen as a continuous fluid, which can be modeled by these aggregated observables only. No inter particle relations are explicitly considered. Given a model, which describes

[3]Islamic pilgramage to Mecca, Saudi Arabia.

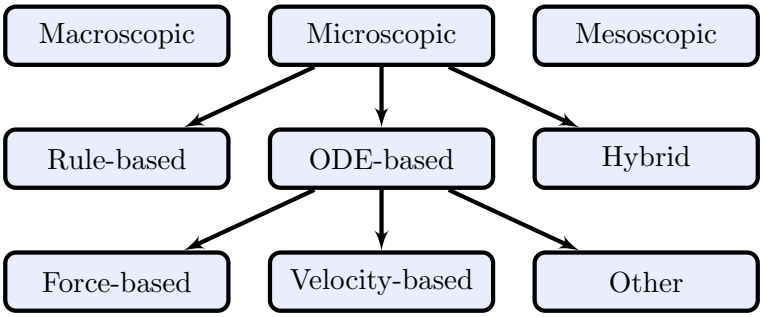


Figure 1.1: A possible hierarchical classification of models in pedestrian dynamics in [4]

the change of the density throughout a geometry, it can be mathematically captured by an PDE. Larger roadmap- and city-traffic-simulation are fields, where macroscopic models are widely spread and can supply travel times and point out bottlenecks [12]. There are limitations to this class of models. They are fast but lack the ability to simulate heterogeneous groups. Nor can they model individual decisions. In an emergency situation, it has been observed, that pedestrians stay in a group, even if other exits are available. This phenomenon is already introduced in microscopic pedestrian models [8][16]. A macroscopic, flow-based model would also have the pedestrians use all available exits [13].

Microscopic models consist of mathematical formulations describing the state and the interactions of every agent. Each agent has a position in the domain and interacts with its environment. It is assumed that the dynamics in any crowd is the result of individual actions. Within the model, these individual actions obviously must be different from the attempt to model the complete, complex system of a person's psychology, which defines its motivation of movement inside a crowd. It is desirable to have few and simple equations to model the agent's motivation. In analogy to Newtonian dynamics, it can be modeled by driving and repelling forces [10]. They lead to second order ordinary differential equations. A popular starting point origins in the modeling of the behavior of electrical charges in an electromagnetic potential field (see figure 1.2). Charges of the same sign act on

each other with a repelling force. This effect is used in the modeling of the natural collision avoidance of a person with other persons, walls and obstacles in pedestrian dynamics. Superposing a driving force, that acts on the agent, steering it towards its destination, is resulting in a *force-based* model and can be described by an ODE.[4] Force-based and velocity-based

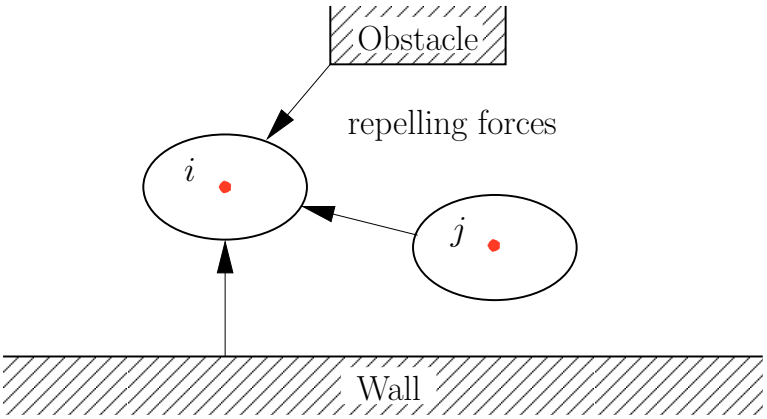


Figure 1.2: Forces acting on agent i from: wall, obstacle and agent j

models are subject of this thesis, so we turn to ODE-based models, which is the super class to both.

1.1 ODE based, microscopic models

We focus on ODE-based microscopic models, which are successful in producing system phenomena like congestions in front of bottlenecks and showing good accordance with experimentally determined fundamental diagrams[5] [7]. Despite the good accordance in mentioned phenomenas, *second order based* models have a weak spot. They tend to show *oscillations* in their trajectories [6]. In a one-dimensional scenario, an agent, approaching a blockade, would oscillate back and forth with decreasing amplitude instead of monotonically decreasing its speed and stop. This oscillation is induced by the system of opposing components of driving and repelling force, which

[4]For further reference, please see corresponding literature, (e.g. [4]).
[5]The Fundamental diagram is the relation of pedestrian flow J and crowd density ρ.

decelerate the agent in the domain, where the opposing force supersedes. Once the agent decelerates and stops, it gets accelerated into the opposing direction until it leaves the domain of supersession (see figure 1.3). Then the deceleration starts over with opposing direction. Various authors try to find new models with enhanced characteristics in terms of directing agents [15][5].

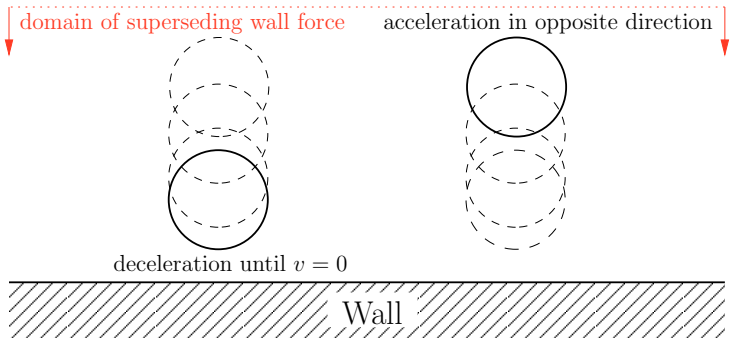


Figure 1.3: Agent gets stopped and forced back by acceleration of wall force.

The difficult calibration of a model is another important issue. This type of models find their limit, if a unique set of parameters for various situations is desired. Best results are achieved with a special calibrated set of parameters for different situations. The change of parameter sets is problematic, if we want to make extensive use of parallel solvers. In terms of ergonomics, a constant set would be more user-friendly.

Velocity-based models, which often lead to first order ODEs, make up an important subgroup. These models change the agent's velocity directly and thus show much better trajectories in terms of oscillation. The Wall-Avoid-Model of this thesis, is partly derived from the *GNM* [7], a velocity-based model. The GNM intends to overcome shortcomings of both groups, oscillation (SFM) and the difficult mathematical treatment of rule-based Models, yet have the positive characteristics remain.

A major step towards this goal is the use of a navigation field, given by the solution to the Eikonal equation. This approach, introduced by Hartmann [9], provides routing and navigation information. In the GNM,

Dietrich divides the navigation into two components. A static and a dynamic navigation vector are described. The static navigation field comprehends the geometry, the dynamic navigation field integrates pedestrians and mobile obstacles. It is clear, that the dynamic navigation field must be computed for every timestep during the simulation.

Besides oscillation and calibration, there is a third issue, on which we will focus in the next chapter: Overlapping. It describes a situation, where an agent's position is invalid either because the agent's simulated presence overlapps with another agent or because it overlapps with a wall or even an obstacle (see figures 1.4, 1.5). Once an agent is fully clipped through a wallsurface, faulty trajectories are most certain.

Figure 1.4: Agents get pushed into obstacles by the large amount of other agents. The agents' colorcode refers to their walking speed. (Red := Stopping; Simulated with a force-based model)

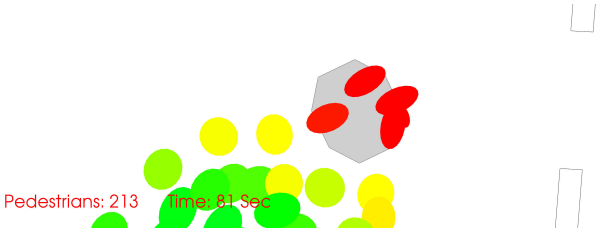


Figure 1.5: Agents remain inside obstacle. (Simulated with force-based model)

These issues highlight the need to develop pedestrian models and search for yet another model, which might overcome some of the shortcomings and can produce as good results as existing models already provide.

1.2 JuPedSim

While designing a new pedestrian model, implementation, testing, calibrating, verification and validation are processes to go through. Next to implementing the model itself, researchers need to calibrate model parameters, run tests, analyze the simulation results and compare them to empirical data as part of the validation process. This chain of steps requires an infrastructure to

- configure simulation inputs, like geometry data, number of agents, agent parameter and of course model parameter in a defined format,
- run a simulation, which produces the trajectories in a defined format,
- verify the model during implementation phase to the specification,
- visualize the simulation results,
- analyze the results and measure metrices like density, flow and compare these to empirical data to validate the model.

If there are no tools for the various tasks at hand, a great amount of time is spent on programming the needed infrastructure. To shift the effort, which goes into the programming of the environment, to the core task, the development of pedestrian models, a framework has been created.

The Jülich Pedestrian Simulator (JuPedSim) is an extensible software framework for simulating and analyzing pedestrians' motion, which supports the user in all of the above phases. It consists of three major modules, a simulation module (JPScore), a visualization module (JPSvis) and a reporting module (JPSreport). The input- and output files are XML based and provide a readable, ergonomic file format. Through JuPedSim, we can focus on our core task, the model formulation and its implementation. We are provided with a ready to use framework, so we can then directly formulate our simulation configurations for the test and calibration process. Results can be visualized with its module *JPSvis*. JuPedSim is platform independent, released under the LGPL license and written in C++.

2 Modelling

2.1 Motivation

In many of second order models, agents breach wall surfaces and get stuck inside of walls. This undesired phenomenon highlights the challenge in calibrating forces and parameters of existing models, so that agents show valid natural behavior while not getting overlapping in extreme situations. Especially in situations of high crowd density, e.g. when facing bottlenecks, overlapping can occur.It leads to inaccuracies in measurements, e.g. in counting and flow calculation.

In the SFM, the walls do have a repulsive force pointing perpendicular to the wall surface. These forces need to be calibrated to work as intended.

Smaller forces might not be strong enough to avoid overlapping with the wall if an agent is in between a wall on one side and many other agents on the other side (see fig. 2.1). The agents on the other side affect that one agent, forcing him towards the wall, while the wall itself acts on the agent in the opposite direction. The resulting force could still point in the direction of the wall, leading to overlapping.

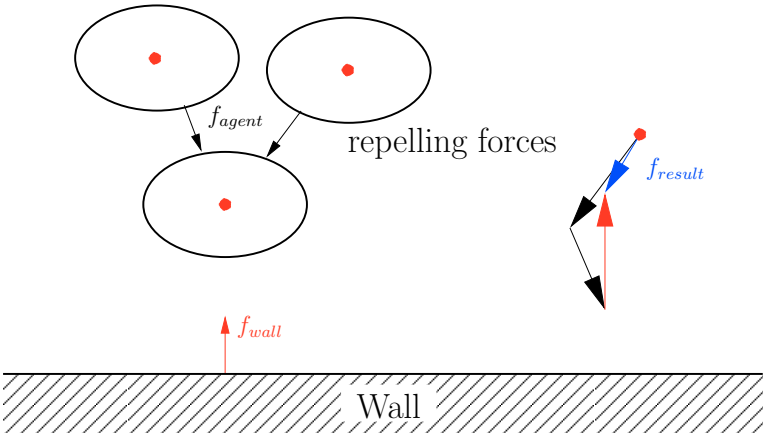


Figure 2.1: Triangle of forces. Small wall-force in red. Sum of all forces in blue.

If the repulsive wallforces are too strong, pedestrians will not use the space close to a wall, even if the domain is very crowded (see fig. 2.2).

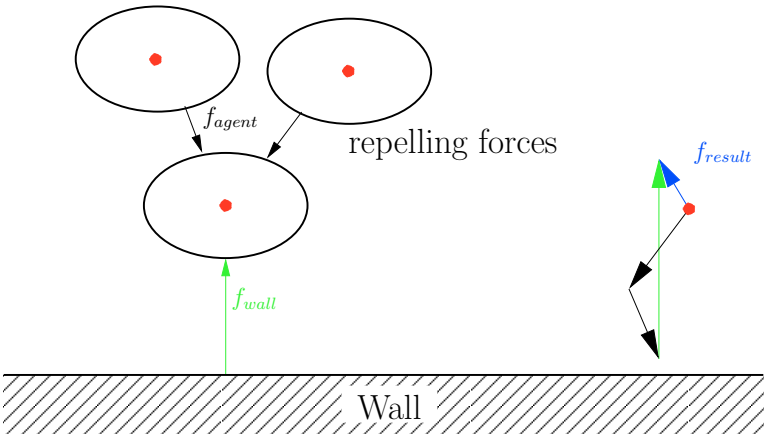


Figure 2.2: Triangle of forces. Large wallforce in green. Sum of all forces in blue.

It is a difficult task, to find a set of parameters, that work as desired in a broad set of situations and geometries. To overcome the problems in calibrating and oscillation, we will define a model based on a first-order ODE. The behavior of agents close to a wall will be the focus. Any mechanic, that will always keep agents away from walls, will not be sufficient. Agents shall keep from walls, only if they have the freedom to do so and are not hindered by surrounding agents. But in situations, where the density is high, agents shall make use of all the available space, even if they get close to walls.

A prototype situation for this scenario to be investigated are the bottleneck-geometries. Here we see a rise in density in front of the point of constriction. After this description of the model behavior, the components of the model will be introduced.

The mechanic to supply an agent with the direction, in which to go, will be a floorfield. It is a field of scalars, that correlates the position in the geometry with a value indicating the time-cost to reach the destination from that position, the *Eikonal Equation*.

2.2 Eikonal Equation

The "Eikonal equation" in a domain Ω, subset of $\mathbb{R}^n$,

$$\begin{aligned} |\nabla c(\vec{x})| &= F(\vec{x}), \quad \vec{x} \in \Omega, \\ \text{s.t.} \quad c|_{\partial\Omega} &= 0, \end{aligned}$$

yields the "time-cost" $c(\vec{x})$ in a spacial domain, provided a target region within the domain as input as well as a slownessfield $F(\vec{x})$. A valid interpretation of "time-cost"-isometrics is to picture a wavefront at a given time t, originating in the target region ($t = 0$) and propagating throughout the spacial domain Ω with the given speed $v = \frac{1}{F(\vec{x})}$ while flowing around any obstacles (see figure 2.3).

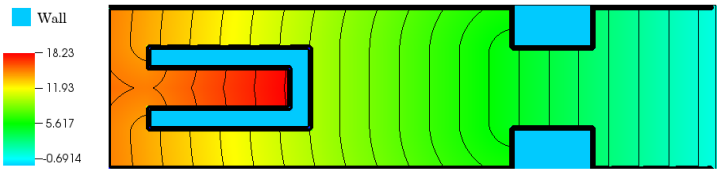


Figure 2.3: Isometrics of "time-cost".

Given a discretization of the domain Ω and the target region $\partial\Omega$, the solution to the Eikonal equation can be approached by using the Fast Marching algorithm [18] (see appendix 5.1). The algorithm provides a first order approximation, yet sufficient for our cause. Computingtime of Fast Marching is independent[6] of the complexity of obstacles and walls.

The negative gradient $-\nabla c$ of the time-cost field is a useful tool in the routing of agents to the target region used as one of the algorithm's inputs. The resulting trajectories show the shortest path to the destination region.

We will refer to the result of the Fast Marching Algorithm as "floorfield".

[6]Fast Marching completion time depends mainly on the length of the wavefronts. If the geometry leads to small lengths, as in geometries with large amounts of narrow corridors, completion time decreases.

To successfully use these floorfields, we discuss and analyze a modification, which gives us smooth trajectories, as proposed in [14].

2.3 Safe Navigation using the Floorfield

When using the plain approximation to the Eikonal Solution, agents anticipate a non smooth pathway that leads very close to walls (see white trajectories in figure 2.7). In many models for pedestrian dynamics, agents, which are very close to walls or obstacles, could overlap with them in rare occasions. They might leave the valid domain and find themselves captured inside walls or obstacles. Yet in reality, we can observe, pedestrians avoiding walls and obstacles with a certain distance.

Therefore, it is desirable to define a modified quality of an optimal route, which accounts for a minimal arrival time and a safe pathway. Safe in respect to avoiding the vicinity of walls and obstacles, whenever possible. It is clear, that altering the floorfield will result in different trajectories. They are no longer the shortest pathways.

In high density situations, the agent shall use all available space. This will be achieved by the way we integrate the floorfield into the velocity vector. If a space is very crowded (high density), then agents should make use of the given space even if that means getting close to walls.

How can an agent “avoid” the close vicinity of any wall or obstacle via an enhanced floorfield?

2.4 Distances-Field

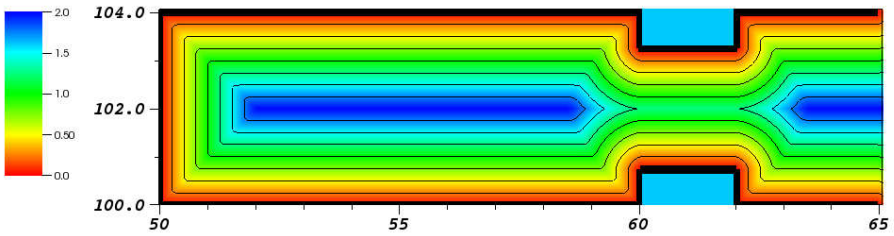


Figure 2.4: Distances field $d(\vec{x})$ of a bottleneck geometry

Having above question in mind, we first need to introduce and understand the *Distances field*, a function d living on the spacial domain Ω, that holds information on how distant the closest wall is. This function will prove useful when altering the one floorfield, we will use for routing. To avoid confusion, let it be emphasized, that this distances field will be used in two different parts of the model. It is used to:

1. create a Direction-to-Wall Field (vector field) and
2. to create a slowness field (scalar field) to initialize the Fast Marching algorithm of the Navigation Field.

The Direction-to-Wall Field is a normalized vector field. Each vector has unit length. The orientation is gained by the negative gradient to the Distances Field. This way, every given vector at $\vec{x}$ directs to the closest wall of that given grid-point $\vec{x}$.

The function value $d(\vec{x})$ will be used to create a slowness field on the discrete grid. This slowness field F holds the value, which determines, how slow the 2-D wave will propagate over the gridpoint in the final *navigation* field. We create a band around walls with the width ω. Only within this band, the wavefront of the Eikonal equation is slowed down. Points, that are within a distance of the ω-band to a wall, have a corresponding low speed

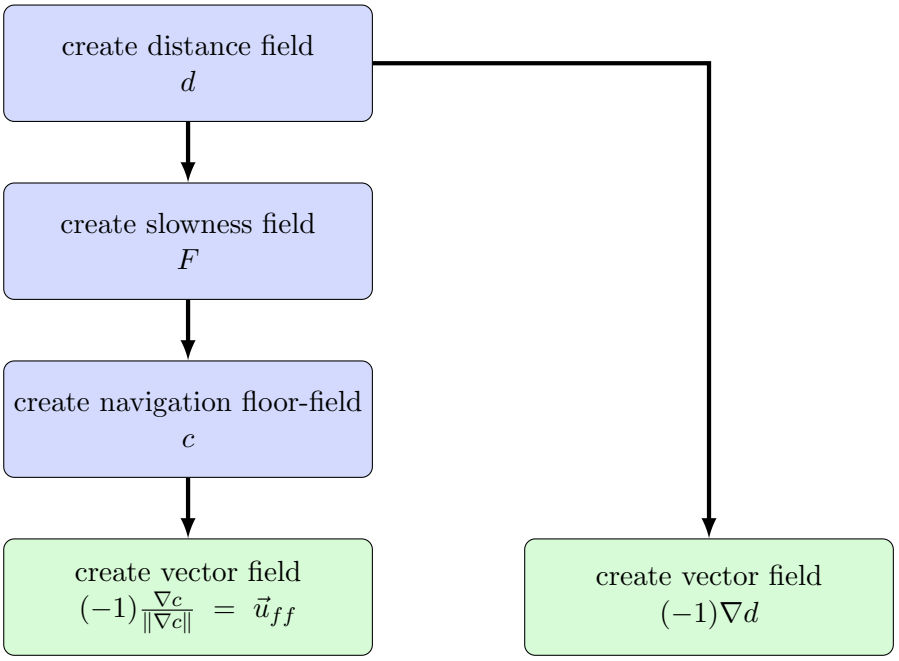


Figure 2.5: Usage of the Distances Field in both vector fields

value $v = \frac{1}{F}$ (see figure 2.6). Routes within the ω-band will take more time

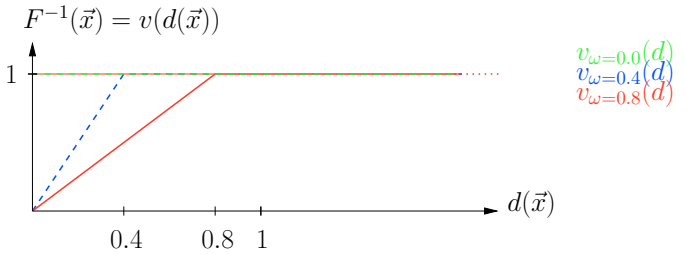


Figure 2.6: Relation of wave propagation speed $v = F^{-1}$ and distances field for different values of ω.

per gridpoint and be less optimal in a sense, that combines travel distances and wall avoidance. We will refer to the value $c(\vec{x})$ as the *time-cost value* (*cost* in short). The effect of this enhancement is seen in figure 2.7.

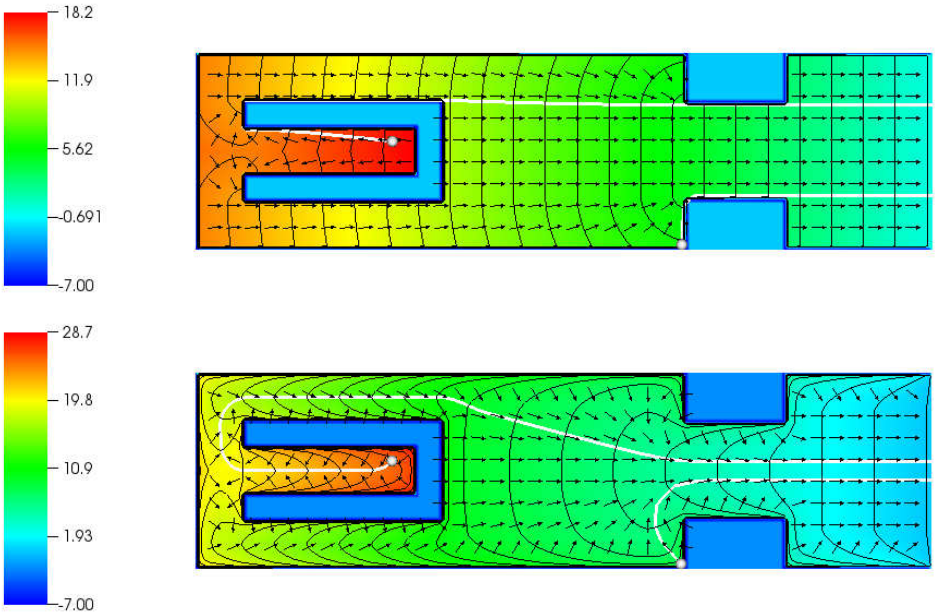


Figure 2.7: Isolines of a floorfield (above) compared to isometrics of the enhanced floorfield (below). Sample trajectories in white.

The vectors of the resulting navigation field close to walls are pointing away from the wall. The effect is best shown by plotting two sample trajectories in both navigation fields. In figure 2.7 above, the agents are routed alongside of walls, whereas below, they avoid walls even if starting close to a wall.

2.5 Wall-Avoid Model

The Wall-Avoid model is velocity-based, using

$$\partial x = \partial t \cdot v(t).$$

The (Euler)-discretization shall be

$$\Delta x = \Delta t \cdot v(t_i),$$

where $v(t_i)$ implements the core of the model, aiming for the avoidance of faulty interaction of pedestrians and walls while maintaining the positive characteristics of reproducing pedestrian crowd behavior (e.g. lane-formation).

There are three mechanics used in the model to avoid "overlapping" in the vicinity of walls:

1. The routing of pedestrians makes use of the Eikonal equation, computed with an *inhomogeneous slowness field*, $F(x)$, whose resulting floorfield[7] favors keeping a distance to obstacles, walls and corners.

2. In a *slowdown-band* the angle between an agent's moving direction and the wallsurface's perpendicular affects the moving speed if and only if the agent's moving vector includes a component geared towards the wall. This is achieved by using the scalar product of the moving direction and the wallsurface's perpendicular.

3. If an agent's distance to a wall drops below a threshold α_r, it is *redirected* to move parallel to the wall if and only if the agent's moving vector includes a component geared towards the wall.

In order to keep the model simple, repulsive wall forces as seen in Social Force Models are omitted. An analogy to repulsive pedestrian forces though is used to keep agents from colliding with each other. The model differs from SFMs, as in SFMs, other agents repulsive forces are transformed into acceleration vectors and from there into a velocity component, which is part of the agent's velocity. We have already seen, that this induces oscillations. In this model though, repulsive influences are not treated as Newton mechanics teaches us, but are only used to factor the repulsive pedestrian effect into a direction component. The speed on the other hand is effected by the other agents only to a certain degree. To show this in the formulation of the model, we used $\vec{i}$ to describe repulsive influences in order to avoid mistaking these components for forces.

[7] see chapter **Eikonal Equation, Safe Navigation using the Floorfield**

Table 2.1: Parameters used:

parameter	value	description
ω	[0.0, 1.6]	threshold for reduced propagation speed (Eikonal)
α_s	0.6	threshold for wall distance, agents get slowed down if closer
α_r	0.3	threshold for wall distance, agents get redirected if closer
α	0.8	weight for linear combination of preceding and current unit speed vector
a	15	asymptotic value of Gompertz function (agent to agent influence)
b	0.25	cut off radius of influence (Gompertz function)
b	3	steepness of Gompertz function

Symbols:

- $\vec{i}$ denotes the influence vector among agents. The correlation of distance and influence is the Gompertz function $f(x) = a \cdot e^{-b \cdot e^{-c \cdot x}}$. The function is smooth and adjustable in its asymptotic value (a), range of influence (b) and steepness, which we can interpret as the elastic modulus of two enclosing agents(c). *Influence* only applies to agents within a view angle $\beta \leq 100\,^\circ$ of the current agent.

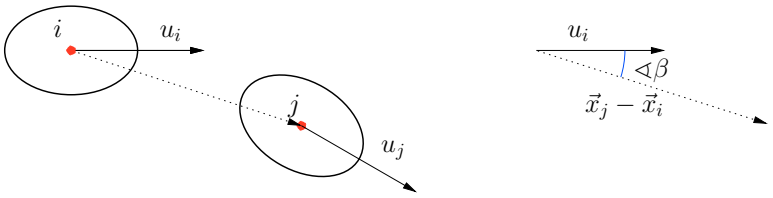


Figure 2.8: Influence taken into account only if agent is catching up to agent in his view. Here, agent j is in view, but i is not catching up. ($\gamma > 90\,^\circ$)

- $\vec{u}$ denotes the *unit speed* vector. At any time, we store the calculated orientation in this vector. It's length is always $\leq 1\,unit$. We call it

unit speed vector, as in most cases, it is the velocity vector with the speed of $1\,m/s$. We multiply it with the agent's *desired speed* value to get the velocity vector of the current agent at the current timestep.

- $\vec{x}$ denotes the position vector of an agent in $\mathbb{R}^2$.
- v_0 denotes the agent's desired speed.

Modul Parameter:

- α denotes the weight of the *unit speed* vector of the previous timestep. $(1-\alpha)$ is the weight of the current *unit speed* vector. (here $\alpha = 0.8$.)
- α_s is the width of the *slow down* band. Any agent, that is closer to a wall than α_s will be slowed.
- α_r is the width of the *redirection* band. Any agent within will be redirected to move parallel to the wall.
- ω is the *wall avoid distance*. It describes the bandwidth around walls, in which the navigation wavefront is slowed down. (see figure 2.6)
- a, b, c are parameters of the Gompertz function and can be set in the configuration file of a simulation.

Functions:

Let Ω be the discrete set of gridpoints in the bounded domain, which holds the geometry of the simulation, a subset of $\mathbb{R}^2$. The following functions will be used in the model formulation and shall be introduced:

$$d : \Omega \ni \vec{x} \longrightarrow d(\vec{x}) \in \mathbb{R}, \qquad \Omega \subseteq \mathbb{R}^2$$

assigns to each gridpoint in Ω the distance to the closest wall. It will be used to choose, in which mode the movement vector candidate will be altered.

(Modes are: regular, slowdown and redirect)

$$P : \mathbb{R}^2 \times \Omega \ni (\vec{u}, \vec{x}) \longrightarrow P(\vec{u}, \vec{x}) \in \mathbb{R}^2$$

describes the orthogonal projection of a given orientation $\vec{u}$ onto the closest wall of $\vec{x}$. It yields an orientation parallel to that wall.

$$u_{ff} : \Omega \ni \vec{x} \longrightarrow u_{ff}(\vec{x}) = \vec{u}_{ff} \in \mathbb{R}^2|_{\|\cdot\|_2=1}$$

is the (normalized) negative gradient of the navigation floorfield at position $\vec{x}$. This vector describes the direction of the negative gradient only and always has unit length. It is used to contribute directly to an agent's unit velocity vector.

$$g : \mathbb{R}^2 \ni \vec{u} \longrightarrow g(\vec{u}) \in \mathbb{R}^2|_{\|\cdot\|_2\leq 1}$$

limits the length of any input vector $\vec{u}$ to unit length. If $\|\vec{u}\|_2 \leq 1$, then g is the identity of $\vec{u}$.

$$\sum_{j=1}^{n} \vec{i}_{rep,j} : \bigotimes_{j=1}^{n} \mathbb{R}_j^2 \ni (\vec{i}_1, ...\vec{i}_n) \longrightarrow \sum_{j=1}^{n} \vec{i}_{rep,j} \in \mathbb{R}^2$$

is adding the distance-dependant[8] repelling influence of neighboring agents within a close vicinity. This sum is analog to the SFM model, yet the resulting influence is directly taken and not converted to an acceleration nor to a velocity. The character i is a hint to being an influence vector, *not* to being a velocity.

Velocity vector $v_0 \cdot \vec{u}_{n,res}$ in timestep n:

[8]The correlation of distance and influence is the Gompertz function $f(x) = a \cdot e^{-b \cdot e^{-c \cdot x}}$. The function is smooth and adjustable in scale, range of influence and steepness by the parameter a,b,c.

$$
\begin{aligned}
\vec{u}_{n,pre} &= \alpha \cdot \vec{u}_{n-1,res} + (1-\alpha) \cdot g \left(g(\vec{u}_{ff}) + g(\sum_{j=1}^{n} \vec{i}_{rep,j}) \right) \\
\vec{u}_{n,res} &= \begin{cases}
(1 - \langle \vec{u}_{n,pre}, -\nabla d \rangle) \quad P(\vec{u}_{n,pre}) \quad \textbf{if} \quad (d(\vec{x}) < \alpha_r) \quad \wedge \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad (\langle \vec{u}_{n,pre}, -\nabla d \rangle \geq 0) \\
(1 - \langle \vec{u}_{n,pre}, -\nabla d \rangle) \qquad \vec{u}_{n,pre} \quad \textbf{if} \quad (\alpha_r < d(\vec{x}) < \alpha_s) \quad \wedge \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad (\langle \vec{u}_{n,pre}, -\nabla d \rangle \geq 0) \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \vec{u}_{n,pre} \quad \textbf{else};
\end{cases} \\
\Delta \vec{x}_n &= \Delta t \cdot v_0 \cdot \vec{u}_{n,res}.
\end{aligned}
$$

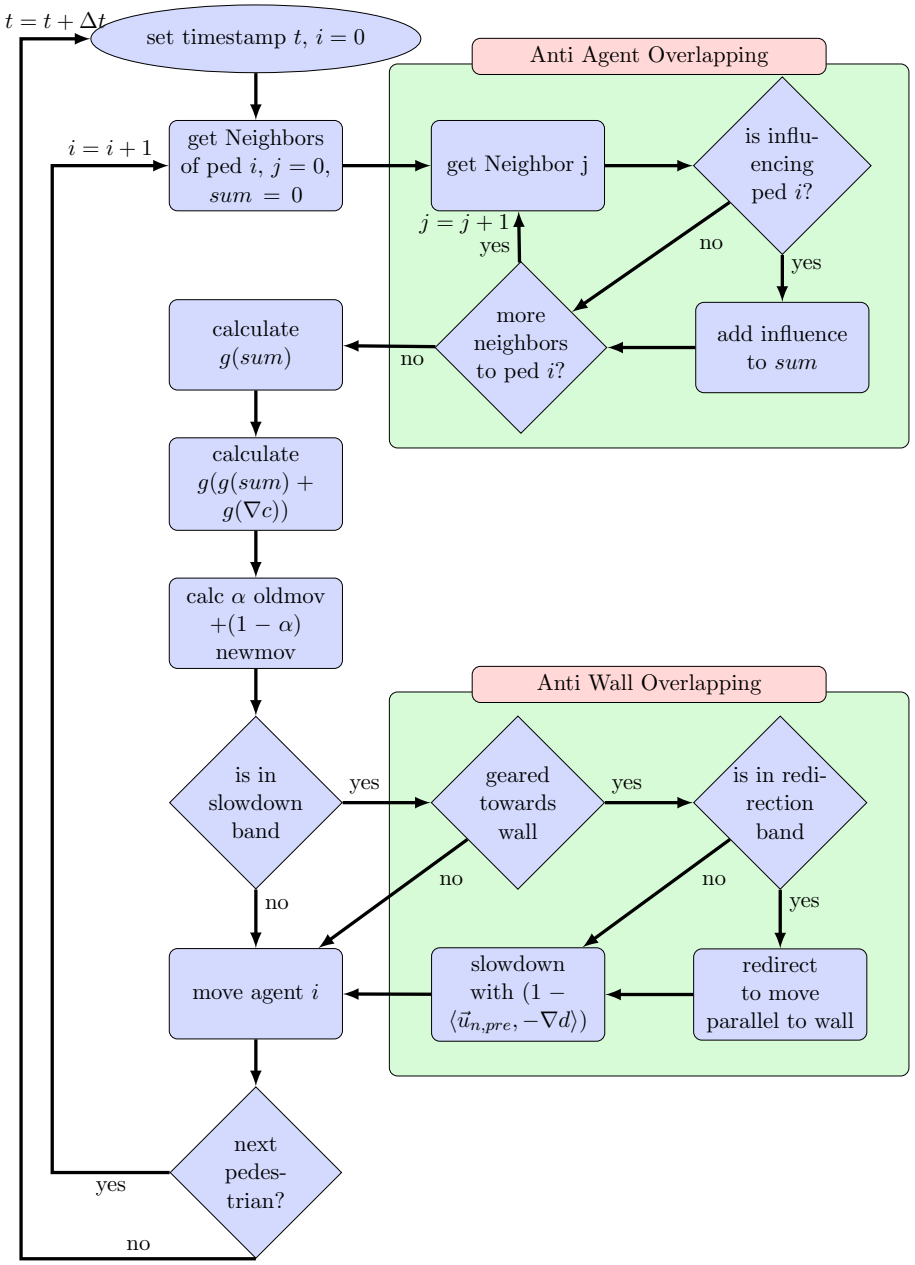


Figure 2.9: Calculation of pedestrian movement during one timestep

What might seem curious at first, is the fact, that both, the navigation field $\vec{u}_{ff}$ and the *sum* of pedestrian influences $\sum \vec{i}_{rep,j}$, are restricted to the length of one unit by function g. Then their sum in turn is restricted again.

This obviously breaks the principle of superposition of forces. We cannot talk about a force-based model here and loose the analogy to Newton's second law the first time applying g on the sum of influences $\sum \vec{i}_{rep,j}$. The resulting vector indicates a new orientation and a slow-down mechanic, as the vector can be of length ≤ 1 unit.

The sum of the navigation field (static) and the accumulated pedestrian forces (dynamic) are restricted by g and then weighted by $(1-\alpha)$, the speed vector of the last timestep gets weighted by α. This is done to reduce flickering[9] of agents. By using the unit speed vector of the preceding timestep, we introduce a discrete approximation of $\partial v = f(t, x, v(x, t))$. It is important to limit that effect. Otherwise we end up with a second-order model. The weighted sum limits the change of orientation and thus an agent could oscillate between two opposing walls. If oscillation should occur in any case, the weight should be shifted away from the preceding timestep. This way, the effect of second-order ODEs gets diminished.

In the next step, we process $\vec{u}_{n,pre}$. The distance to the next wall is checked and if the agent is closer than α_s to any wall, we calculate the scalar product $\langle \vec{u}_{n,pre}, -\nabla \hat{d} \rangle$ to evaluate, if there is a component towards the closest wall. In this case, the agent is slowed down (see figure 2.10). If

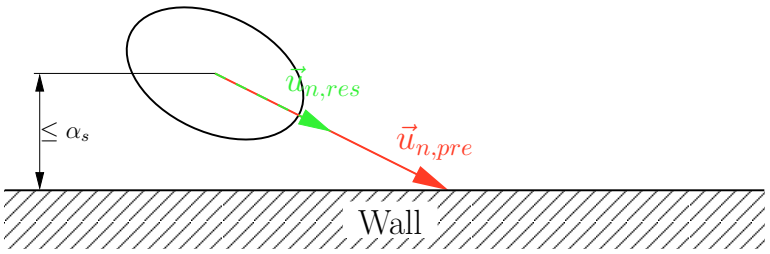


Figure 2.10: Slowdown: Reduction of the unit speed vector to avoid clipping

the agent is already very close to the wall, $(d \leq \alpha_r)$, the velocity component

[9] We call the change in direction back and forth in every timestep *flickering*. We want to shortly address a second alternative approach. If one is willing to accept flickering agents with fast changing orientations, one could omit the velocity vector of the timestep $(n-1)$ and postprocess the trajectories. As we only get positions at discrete timesteps, one could easily create a smooth trajectory by using *B-splines*.

towards the wall gets neglected. Thus, the agent gets directed to move parallel to the wall (see figure 2.11). α_s and α_r define the widths of two bands, in which we alter the vector $\vec{u}_{n,pre}$ (*pre* for prediction) to get the final unit speed vector $\vec{u}_{n,res}$ (*res* for resulting). The bands are arranged as pictured in figure 2.12.

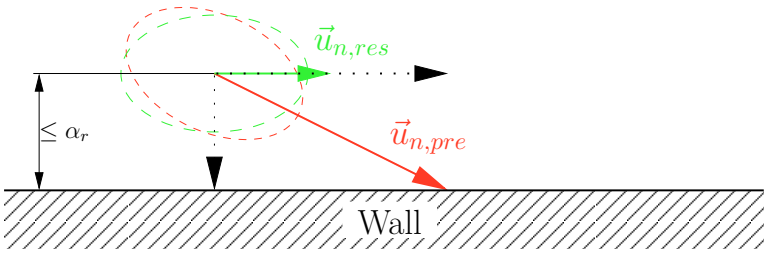


Figure 2.11: Redirection of the unit speed vector to avoid clipping

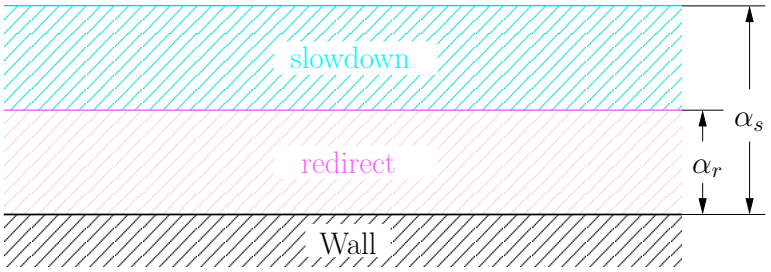


Figure 2.12: Slowdown and redirection band alongside a wall.

3 Verification and Validation

In this chapter, we want to describe applicable tests to the Wall-Avoid model. First, we validate basic fundamentals any pedestrian model should pass. Further, it is verified, that we see no overlapping and that the crowd shows typical phenomena, like lane-formation [20]. Following our set of tests, we will then validate the Wall-Avoid model to produce the pedestrian flow in bottleneck experiments, corresponding to empirical data. This verification is widely considered the most important criterion to validate simulation results [19]. We will further compare the trajectories with a different model, one using automated triangulation to assist routing.

3.1 Basic Tests

The validation of a model is a complex task and makes up a separate research field. Researchers in the field of Civil Engineering are working on various approaches on how to validate a model. The research group, CST - Pedestrian Dynamics and Traffic Simulation, is developing a set of testcases any serious model should aim to pass. Tests include the behavior of a single moving agent passing static objects like a dummy agent or an obstacle.

The Wall-Avoid model, implemented in its current state in the simulation suit JuPedSim[1], passed these applicable tests and it was shown, that the basic mechanics of the routing are working as specified. Yet to complete validation of the model on a more complex level, the floorfield needs to support multiple destinations. This would enable test with bidirectional flows and more.

3.1.1 Test A

In the first test, we see a corridor, a moving agent and a dummy agent positioned in the line of sight to the destination (figure 3.1). This test is passed, if the moving agent will pass the dummy agent without overlapping or tunneling.

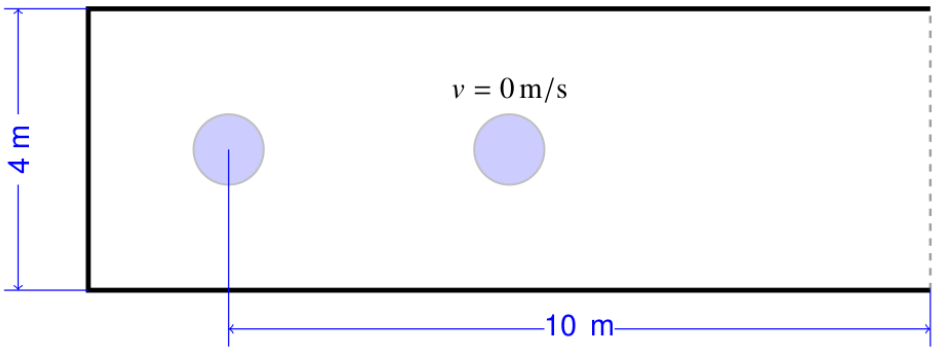


Figure 3.1: Testcase 1: passing a static dummy agent.

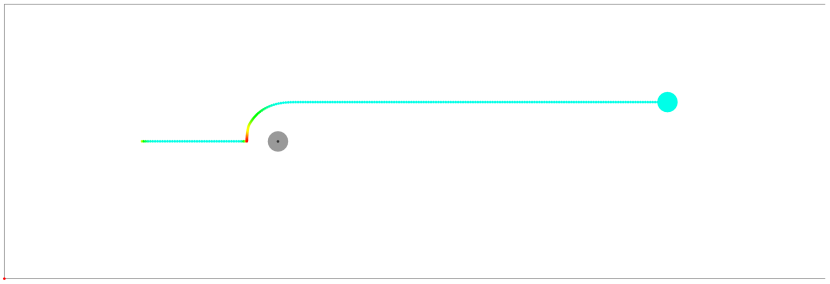


Figure 3.2: Simulation Result: Agent is passing the dummy agent.

As figure 3.2 shows, the agent passed the dummy agent. As the dummy agent is not defined in the geometry, it is not considered in the calculation of the floorfield. The composition of floorfield and agent-to-agent influence leads to this trajectory. The moving agent is not anticipating the dummy agent.

3.1.2 Test B

In the second test (see figure 3.3), the corridor is narrowed down, so that there is not enough space to pass the static agent on either side. The test is passed, if the moving agent will be unable to pass the dummy agent.

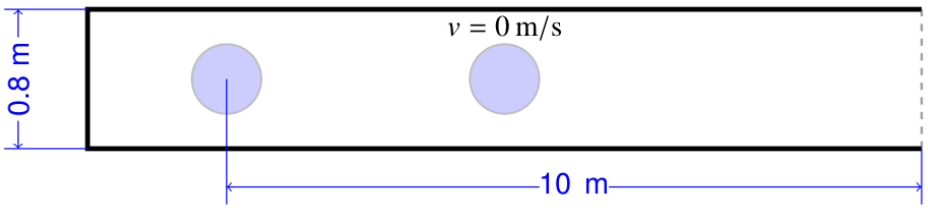


Figure 3.3: Testcase 2: no passing of a static dummy agent.

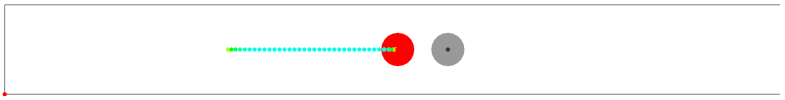


Figure 3.4: Simulation Result: Agent stops before the dummy. Red color indicates $v = 0$.

Simulation Result (see figure 3.4) show, that the test is passed. The agent stops as specified.

3.1.3 Test C

The last of the basic tests describes a situation as depicted in figure 3.5. The destination of the agent is out of sight. The test is passed, if the agent can reach the exit on the left side. In the plot of the simulation

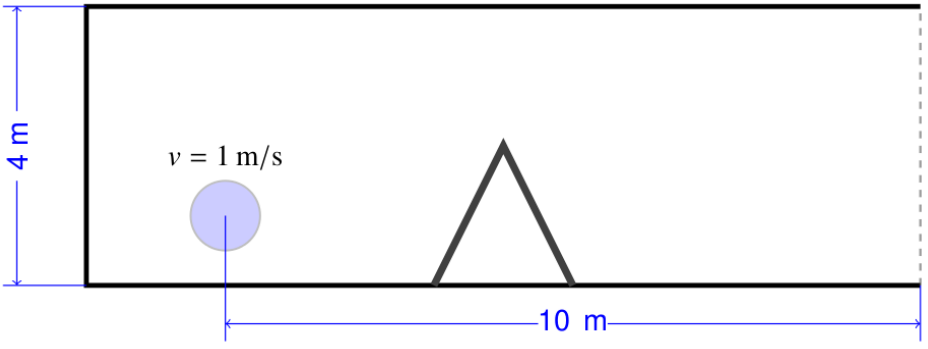


Figure 3.5: Testcase 3: reaching a goal out of sight.

result (figure 3.6) we see, that the agent can anticipate the static object

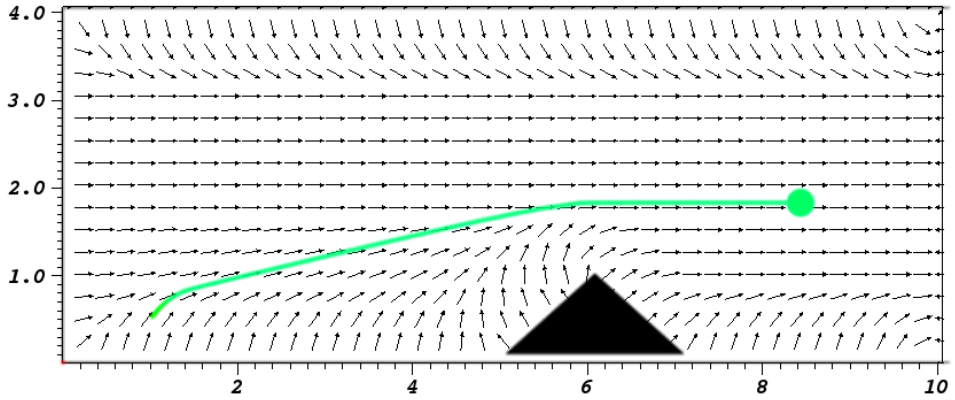


Figure 3.6: Simulation Result: Navigation Field and trajectory of the agent.

in advance. The smooth trajectory shows the capability of the enhanced floorfield to anticipate obstacles in advance.

3.2 Variation of the Parameter

The following test analyzes the *wall-avoid-distance* ω and proves the effective treatment of agent-wall interaction. We simulate bottleneck experiments with bottleneck widths $0.8\,m \leq w \leq 2.5\,m$. Each geometry is tested with $\omega = 0.0$, which results in the plain floorfield and with $\omega > 0.0$, which results in enhanced floorfields. In figure 3.8, we see snapshots of the test simulation. Without the enhanced floorfield, many agents walk alongside the walls, both in the entering-section before the bottleneck and within the bottleneck. At the time of the snapshot, agents already crowd before the bottleneck and use all available space. This usage of available space is seen in the simulation with $\omega > 0$ also. Right after the bottleneck, agents walk right towards the exit when using the plain floorfield (above). The other simulations (middle and below) show the effect, that agents will avoid the vicinity of walls, if they are not hindered and pedestrian density is low. This wall avoidance is also seen in the first phase of the experiments. The first agents entering the bottleneck have the possibility to walk to the middle of the geometry. The more we increase ω, the more we can see the effect of a spearhead (see figure

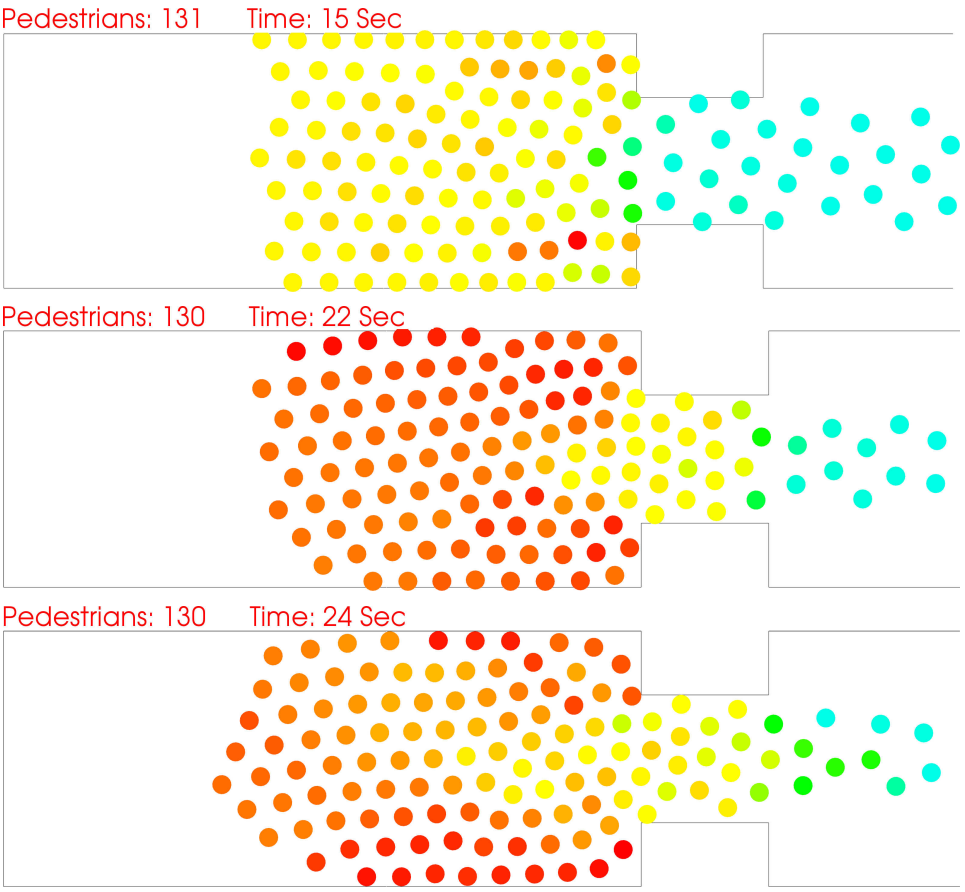


Figure 3.7: Bottleneck Experiment:
above: width: $2\,m$, $\omega = 0.0\,m$
middle: width: $2\,m$, $\omega = 0.8\,m$
below: width: $2\,m$, $\omega = 1.6\,m$

). We get a better look at this effect, when plotting all trajectories as seen in figure 3.9. The effect of the parameter ω can be seen. The bottleneck reduces the pedestrian speed in front of it. If ω is chosen to high, the agents avoid the walls after the bottleneck. Seen in the trajectories narrowing down even further.

The reduction of the flow can be seen in figure 3.10, as w decreases. After the bottleneck, the pedestrians can accelerate again. Figure 3.10 shows the results of experiments, conducted by Kretz in 2006 and Seyfried in 2009. The comparison also validates the model to yield a fundamental diagram like

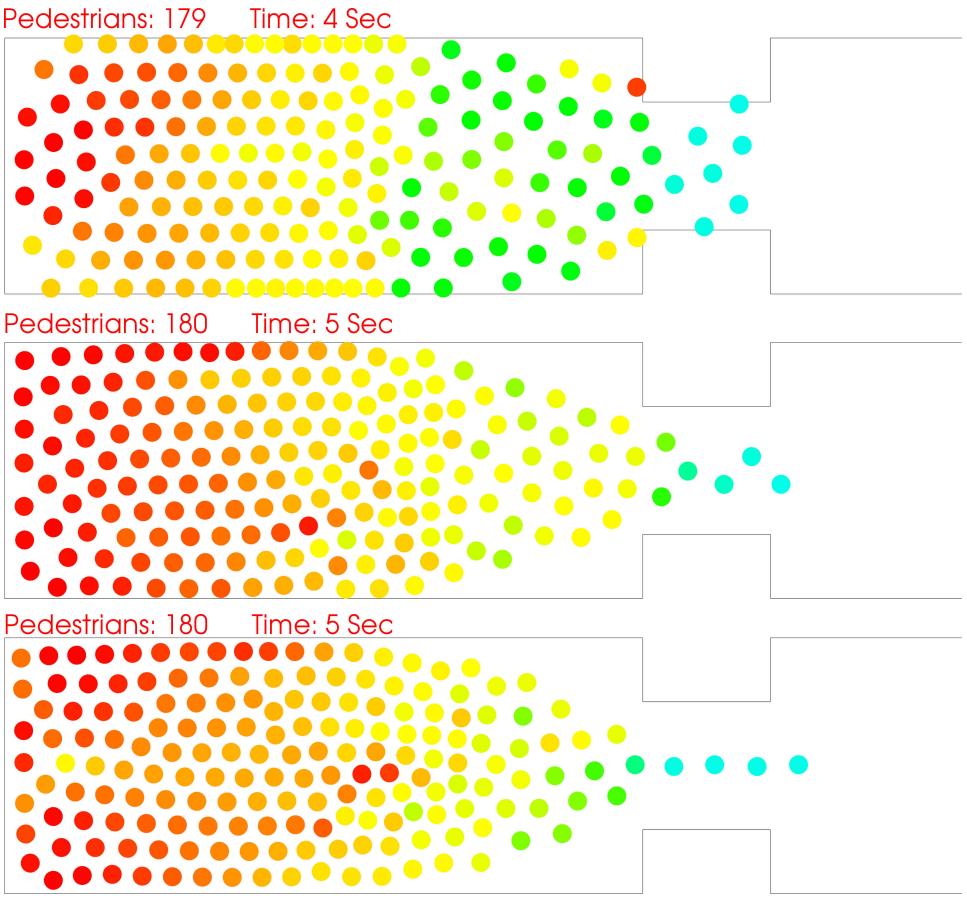


Figure 3.8: Bottleneck Experiment:
above: width: $2\,m$, $\omega = 0.0\,m$
middle: width: $2\,m$, $\omega = 0.8\,m$
below: width: $2\,m$, $\omega = 1.6\,m$

seen in real world experiments: The calculated flow through the bottleneck shall match the empirical data. It can be seen, that the more the agents will strive to avoid obstacles and walls, the lesser the flow will be (see 3.10).

The simulation result also shown, that without the enhancement of the floorfield, few agents do *not* pass the bottleneck, but get caught inside of walls (see figure 3.11).

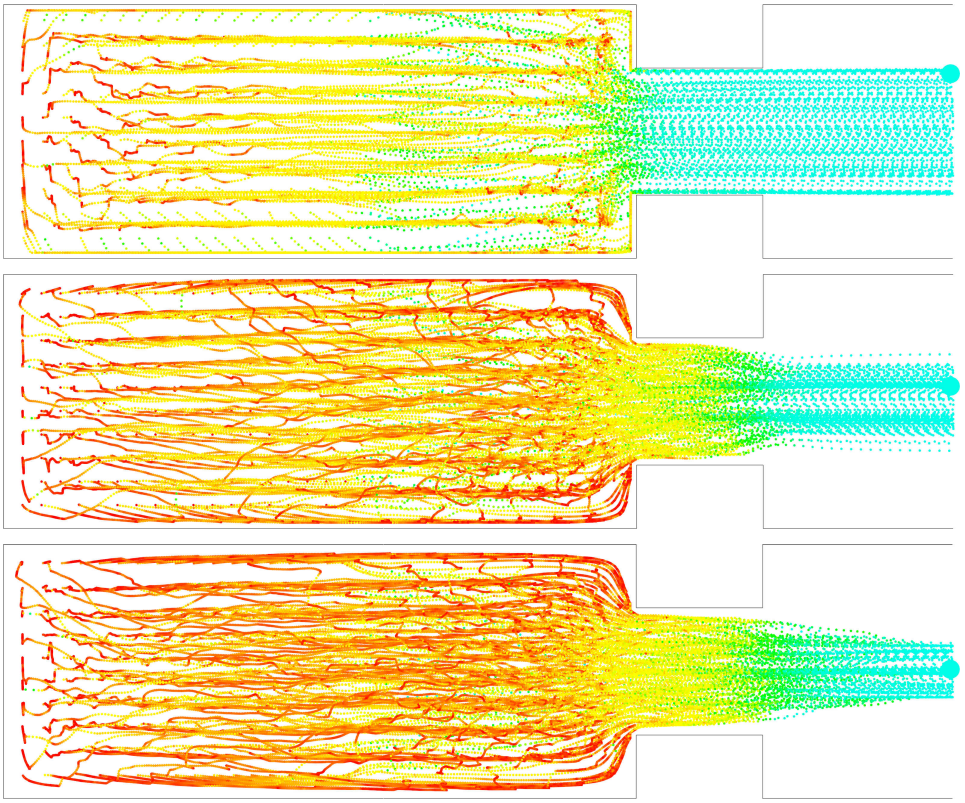


Figure 3.9: Bottleneck Experiment:
above: width: $2\,m$, $\omega = 0.0\,m$
middle: width: $2\,m$, $\omega = 0.8\,m$
below: width: $2\,m$, $\omega = 1.6\,m$

3.3 Compare Trajectories to an existing automated model

The title of this section is surely forcing some questions. What does the author mean by *automated* models. In the context of this thesis, we distinguish among models that need manual alteration of a specified geometry input, e.g. domain decomposition, for the router to function and models that only need the geometry data (see Abstract). If we only allow models of the later, we can transform models of the first class to an automated model by using a form of automated decompositor. In the following, a model that requires the line of sight for an agent to reach an intermediate goal, the manually helplines get replaced by a triangulation. So now we can compare two mod-

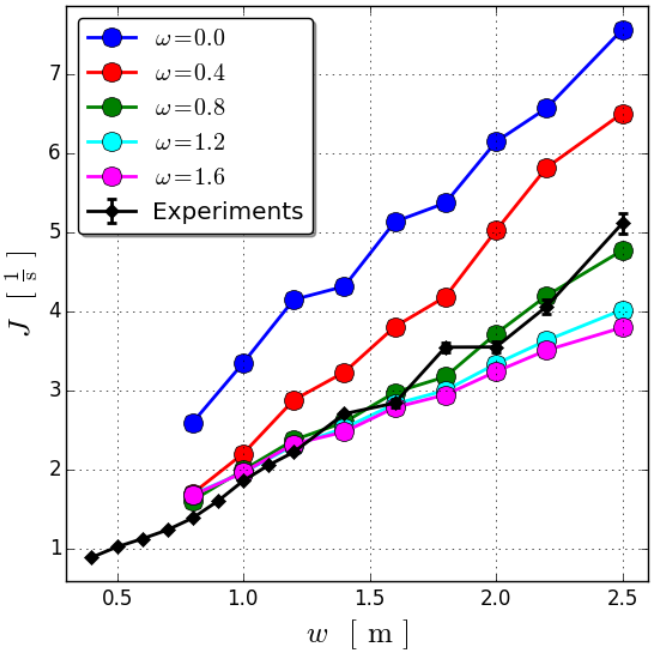


Figure 3.10: Diagram with empiric data and simulation results of various Bottleneck widths.
Data from: 2006 EG, 2006 Kretz, 2009 Seyfried.

els of the same class, and qualify the trajectories of both simulations. (see fig. 3.12) It is seen, that the trajectories of the new Wall-Avoid model have an organic look. They don't have the zig-zag look, seen in the left snapshot (figure 3.12). It is hard to quantify that organic effect, so the qualitative comparison shall suffice.

3.4 Cost of a "full" preprocessing step

Close to all time of the needed computation spent on the enhanced floorfield, the prohibition of overlapping, is spent in a preprocessing step before the actual simulation starts and therefore does not effect the real-time factor. In this Wall-Avoid Model, the preprocessing is increased compared to a *GNM*. Where Dietrich uses the Eikonal solution once (mollifiers not considered), the Wall-Avoid Model uses two runs through the Fast Marching algorithm

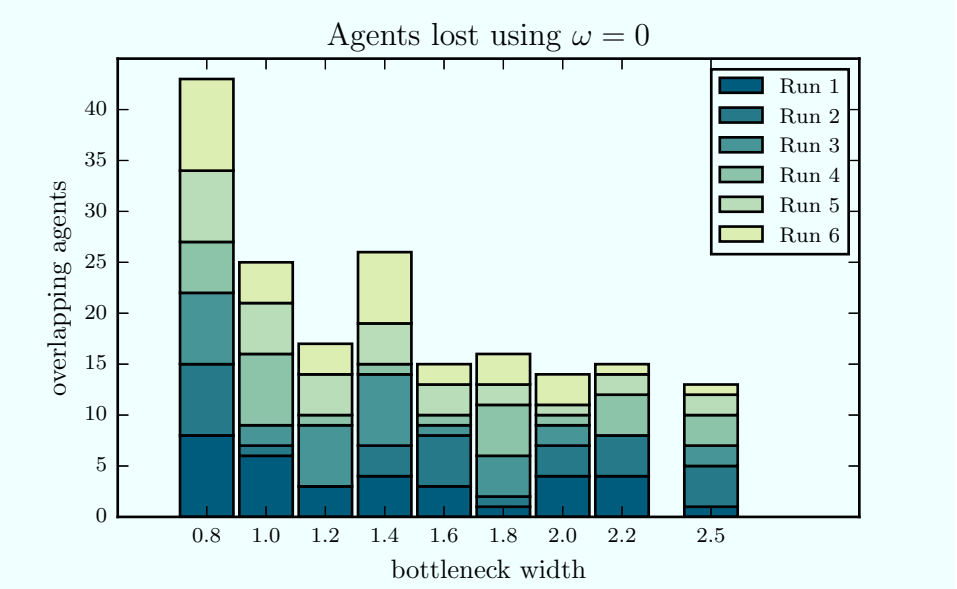


Figure 3.11: Agents lost by overlapping with plain floorfield.

Figure 3.12: Comparison between trajectories: Routing with helplines through triangulation (left) to routing with navigation field (right).

(FMA). In this chapter we want to elaborate on the doubled effort we spend.

The granularity of the rectangular grid governs the cost of the FMA[10]. For pedestrian navigation, we chose a point-to-point distance of neighboring gridpoints of $0.0625\,m$. A geometry spanning $100 \times 100\,m^2$ can be processed

[10] Fast-Marching

in 37 seconds.[11] Including storing all floorfields and gradient fields, processing time rises to 65 seconds.

The Fast Marching algorithm is that fast, that it basically can be neglected when rating the performance. A prequel Fast Marching run does not change that verdict. The usage of the Direction-to-Wall Field in the model shows no more overlapping and seems easily worth the cost. During runtime, using the floorfield or the Distance-to-Wall field means reading a vector and performing up to 2 scalar products.

To show the performance of the Wall-Avoid Model, we started a simulation in a complex geometry with more than 3000 agents (see figure 3.13). Compared to other models available in *JuPedSim* [1], we could not see any significant performance difference. The trajectories improved and look much more natural. This was achieved without the need to manually adding decomposing helplines or intermediate navigation goals.

[11] Measured on a single core of a Intel© Core™ i7-3610QM CPU @ 2.30GHz

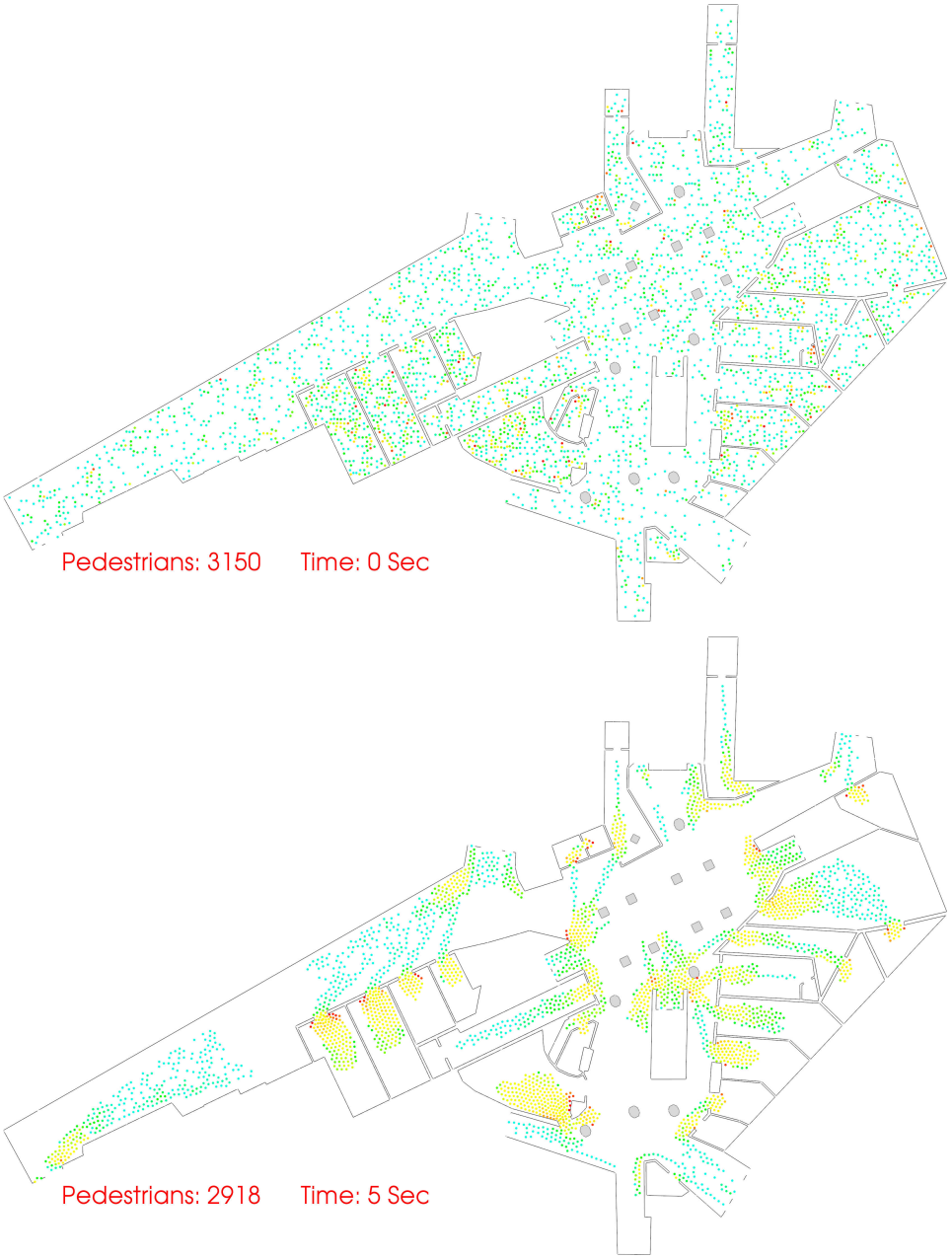


Figure 3.13: Simulation of a complex geometry with multiple exits and 3000 agents

4 Conclusion and Outlook

4.1 Conclusion

In this thesis, an altered usage of an enhanced floor field was shown, integrated in a suitable new Wall-Avoid Model. The *wall-avoid distance* parameter was analyzed and shown to be able to show good convergence to empirical data. During the course of the work process, it became clear how versatile and powerful floorfields can be. Not only in the current state, they can provide valuable assistance to pedestrian models, but they can be further developed to fit into many more contexts.

The parameters of the model are calibrated as follows: The wall-avoidance parameter $\omega = 0.8\,m$ shows best convergence with empirical data as well as the most natural behavior, when visualizing the simulation results. The bandwidth of the slowdown-band is chosen to be $\alpha_s = 0.6\,m$, the redirection bandwidth is $\alpha_r = 0.6\,m$.

4.2 Floorfield

4.2.1 Multiple Goals

The floorfield is a useful tool in routing of pedestrians through any geometry. To unfold its full power, one can advance to calculate a floorfield for each of many atomic goals[12]. The combination of many floorfields, each corresponding to an atomic goal, can easily be managed by selecting the direction vector of the one floorfield, that provides the minimal time-cost of an active set of floorfields evaluated at the gridpoint of the current agent's position. Dynamic world events in a simulation could alter the set of active floorfields. This way, models can implement navigation in a dynamically changing world. It can be a tool in a simulation suit, which has agents change their destination during runtime. This would be realized by simply changing the active set to the floorfields corresponding to the new situation.

[12] An atomic goal would be a single exit door, whereas a destination could consists of a set of doors/exits.

4.2.2 Multiple Floors

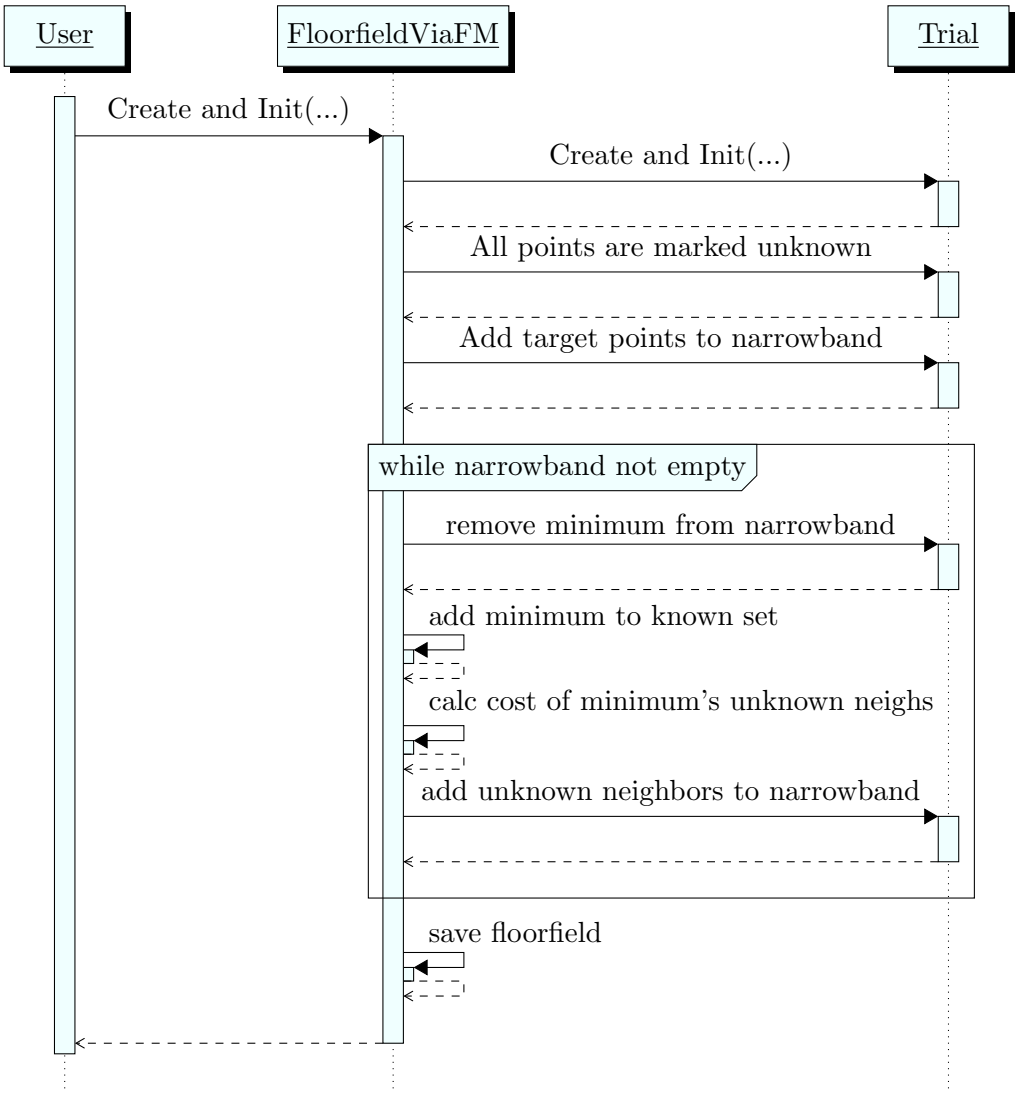
In the current state, the floorfield provides time-cost on a discrete grid, a rectangular grid with equidistant spacing in each dimension. The gridpoints are stored in a one dimensional array by the row-major order. In an arrangement like this, it is easy to formulate 4-neighboring[13] relations. These values are easily accessed, if you are provided the stride value, namely how much gridpoints make up the length of both dimensions in a 2-D world. The Fast Marching algorithm needs the time-cost values of the 4-neighborhood. This will change, if you need to simulate in a building with multiple floors, which are connected via stairs. We need to introduce a new mechanic, which can be treated equally handy inside each floor. Any position, projected onto the x-y plane, may not be unique anymore. On the other hand, it would be a waste of memory, if a third dimension would be introduced[14] in that way, to represent the hull cube circumscribing a building. One must find a solution, which describes the geometry of the rooms in a memory efficient way and yet be able to comfortably access the 4-neighbors' time-cost value.

[13] Gridpoints north, south, east and west to the current are called 4-neighbors.

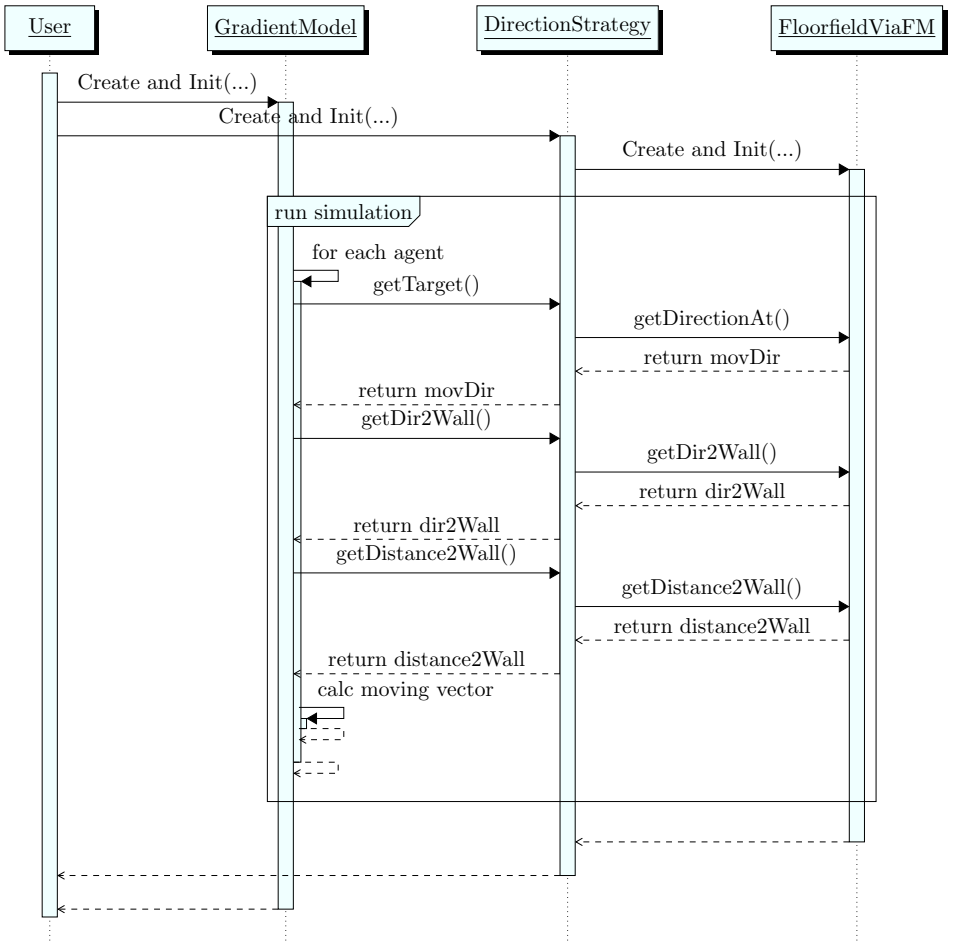
[14] As we are interested in points representing the floor of a room, all the volume (air) above would be not used.

5 Appendices

5.1 Fast Marching Algorithm



5.2 Gradient Model using a Floorfield



References

[1] Jupedsim. http://www.jupedsim.org.

[2] BBC. Hajj stampede: What we know so far www.bbc.com/news/world-middle-east-34357952, 2015.

[3] L. Benedictus. http://www.theguardian.com/world/2015/oct/03/hajj-crush-how-crowd-disasters-happen-and-how-they-can-be-avoided hajj crush: how crowd disasters happen, and how they can be avoided, 2015.

[4] M. Chraibi. *Validated force-based modeling of pedestrian dynamics*. PhD thesis, Universität zu Köln, March, 15 2012.

[5] M. Chraibi, U. Kemloh, A. Seyfried, and A. Schadschneider. Force-based models of pedestrian dynamics. *Networks and Heterogeneous Media*, 6(3):425–442, 2011.

[6] M. Chraibi, A. Seyfried, and A. Schadschneider. Generalized centrifugal force model for pedestrian dynamics. *Physical Review E*, 82:046111, 2010.

[7] F. Dietrich and G. Köster. Gradient navigation model for pedestrian dynamics. *Arxiv e-prints*, 2014.

[8] J. Ding, X. Ling, H. Huang, and I. Takashi. Herding effect in coupled pedestrian-pedestrian interacting dynamics. *Chinese Physics Letters*, 28(12):128301, 2011.

[9] D. Hartmann. Adaptive pedestrian dynamics based on geodesics. *New Journal of Physics*, 12:043032, 2010.

[10] D. Helbing. Traffic and related self-driven many-particle systems. *Rev. Mod. Phys.*, 73:1067–1141, 2001.

[11] K. Hirai and K. Tarui. A simulation of the behavior of a crowd in panic. In *Proc. of the 1975 International Conference on Cybernetics and Society*, pages 409–411, San Francisco, 1975.

[12] INRO. http://www.inrosoftware.com/en/products/emme/, 2015.

[13] P. Marno. Crowded - macroscopic and microscopic models for pedestrian dynamics, 2002.

[14] L. Moreno. Fast marching, roboticslab.uc3m.es/roboticslab/research/fast-marching, 2014.

[15] M. Moussaïd, D. Helbing, and G. Theraulaz. How simple rules determine pedestrian behavior and crowd disasters. *PNAS*, 2011.

[16] G. J. Perez and C. Saloma. Allelomimesis as escape strategy of pedestrians in two-exit confinements. *Physica A: Statistical Mechanics and its Applications*, 388(12):2469–2475, jun 2009.

[17] W. M. Predtechenskii and A. I. Milinskii. *Personenströme in Gebäuden - Berechnungsmethoden für die Projektierung.* Verlagsgesellschaft Rudolf Müller, Köln-Braunsfeld, 1971. Original in Russian, Stroiizdat Publishers, Moscow, 1969.

[18] J.A. Sethian. *A Marching Level Set Method for Monotonically Advancing Fronts.* Proc. Nat. Acad. Sci., 1996.

[19] A. Seyfried and A. Schadschneider. Fundamental diagram and validation of crowd models. In H. Umeo, S. Morishita, K. Nishinari, T. Komatsuzaki, and S. Bandini, editors, *Cellular Automata*, volume 5191/2008 of *Lecture Notes in Computer Science*, pages 563–566. Springer Berlin / Heidelberg, 2008.

[20] Q. Zhang and B. Han. Simulation model of pedestrian interactive behavior. *Physica A*, 390:636–646, 2011. Article in Press, Uncorrected Proof.