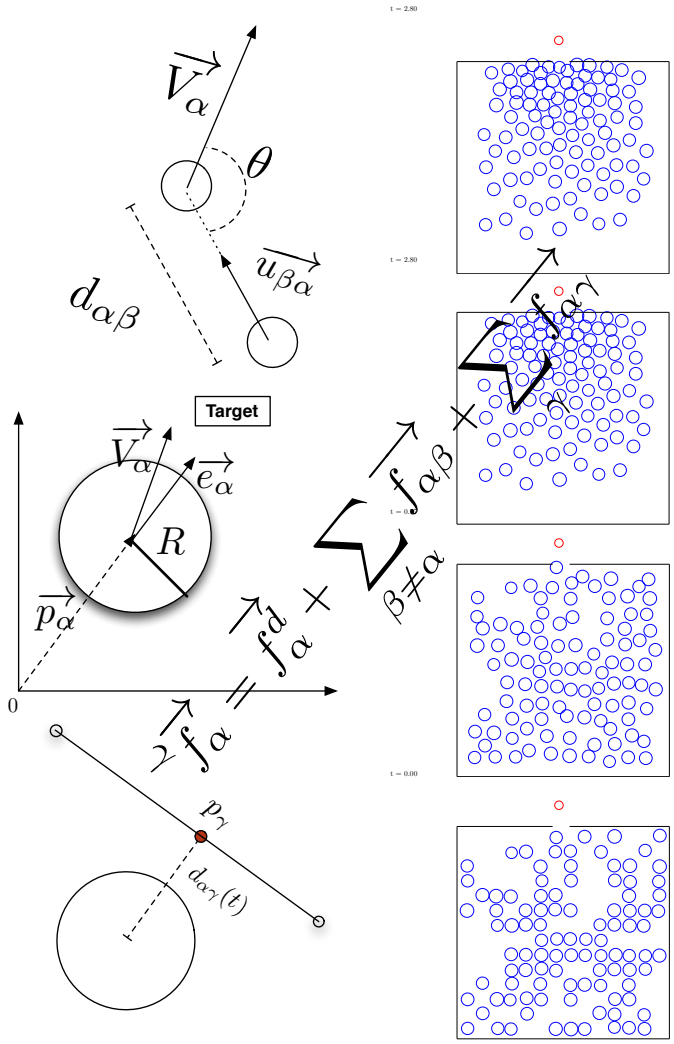
Crowd modelling



Toke Høiland-Jørgensen
Mikkel Hartmann
Dan Albrechtsen
Malik Thrane
Troels Christensen
Wence Xiao

Advisor
Viggo Andreasen

Modelling project, fall 2010
Mathematics RUC

Abstract

We look at social force models as a way to model the behaviour of human crowds, in order to evaluate how well these types of models simulate crowd behaviour, and what the models' strengths and weaknesses are. In order to do this evaluation, we implement a computer simulation of an exemplary social force model.

In order to create this simulation, we pick an exemplary model that is well described in the article that presents it, and analyse it in detail, filling in details from other articles when necessary. Based on this analysis of the model, we go from the abstract model formulation to a concrete numerical simulation by filling in required details, such as how to approximate the movement of pedestrians, how to set initial conditions and values, and how to implement the interaction between pedestrians and walls in practice.

From our results, it is clear that our simulation (with the right parameters) exhibits reasonable pedestrian behaviour upon visual inspection. While we successfully replicate some results from the literature, other effects do not manifest themselves. We discuss several reasons for this discrepancy, including features that are missing from the model, parameter values, effects of using random numbers to generate the initial conditions and possible errors in our implementation of the model.

Based on the results of our own simulations and our review of the social force modelling field, we assess social force models and their strengths and weaknesses. We conclude that social force models are not based on any theories for the behaviour of crowds, but are created to replicate a set of observations. As such, any confidence in their predictions must come from a record of producing results fitting observations; and since the field is relatively new, they have not yet reached this state. Social force models do, however, provide a practical way to simulate something that would otherwise be impossible to simulate. As such, they are the best available way to provide e.g. guidance when designing facilities that must accommodate many pedestrians, and given time the accuracy of their predictions will probably increase.

Resume

Vi ser på "social force"-modeller som et værktøj til at modellere menneskemængders opførsel, for at vurdere hvor godt denne type af model simulerer menneskemængders opførsel, og hvad modellernes styrker og svagheder er. For at udføre denne vurdering, udarbejder vi en computersimulering af en eksemplarisk social force-model.

For at udarbejde denne simulering, udvælger vi en eksemplarisk model som er velbeskrevet i den artikel der præsenterer den, og analyserer den i detaljer. Herunder udfylder vi hvor det er nødvendigt huller i modelformuleringen vha. dele fra andre artikler. Baseret på denne analyse af modellen, går vi fra det abstrakte formulering af modellen til en konkret nummerisk simulering ved at udfylde nødvendige detaljer, såsom hvordan vi tilnærmer fodgængernes bevægelse, hvordan vi opstiller begyndelsesbetingelser, og hvordan vi udregner samspillet mellem fodgængere og vægge i praksis.

Fra vores resultater er det klart, at vores simulering (med de rette parametre) udviser rimelig fodgængeradfærd vurderet ved visuel inspektion. Mens det lykkes os at genskabe nogle af resultaterne fra litteraturen, er der andre effekter der ikke optræder. Vi diskuterer en række årsager til denne forskel, herunder manglende features i modellen, indstilling af parametre, effekter af at anvende tilfældige tal til at generere begyndelsesbetingelser samt eventuelle fejl i vores implementering af modellen.

Baseret på resultaterne af vores egne simuleringer og vores gennemgang af forskningsområdet, vurderer vi social force-modeller og deres styrker og svagheder. Vi konkluderer at social force-modeller ikke bygger på nogen teori for menneskemængders adfærd, men er skabt for at lave simuleringer svarende til konkrete observationer. Som sådan skal tilliden til deres forudsigelser opbygges ved at sammenholde simuleringer med observationer, og da feltet er relativt nyt, har modellerne endnu ikke opnået stor sikkerhed i deres resultater. Social force-modeller giver dog en praktisk måde at simulere noget som ellers ville være umuligt at simulere. Som sådan, repræsenterer de det bedste værktøj der eksisterer til fx at vejlede ved udformningen af faciliteter der skal rumme mange fodgængere, og pålideligheden af resultaterne vil formentlig forbedres med tiden.

Contents

1 Introduction **5**
- 1.1 Problem formulation . . . 5
- 1.2 Target audience . . . 5
- 1.3 Approach to answering the problem formulation . . . 6

2 Social force models **7**
- 2.1 Variants of social force models . . . 7
- 2.2 Results obtained from the models . . . 8
- 2.3 Simulation scenarios . . . 9
- 2.4 Summary . . . 9

3 The model **10**
- 3.1 The desired movement force . . . 11
- 3.2 Interaction force between pedestrians . . . 13
- 3.3 Repulsion from the walls . . . 14
- 3.4 Summary . . . 16

4 From model to simulation **17**
- 4.1 Approximating pedestrian movement . . . 17
- 4.2 Initial conditions and constants . . . 18
- 4.3 Finding the nearest point on a wall . . . 20
- 4.4 Summary . . . 22

5 Simulation approach **23**
- 5.1 Structure of the program . . . 23
- 5.2 Setting up the simulation . . . 23
- 5.3 Calculation of the model . . . 24
- 5.4 Obtaining results . . . 29
- 5.5 Summary . . . 30

6 Results **31**
- 6.1 The basics . . . 31
- 6.2 The results from the literature . . . 32
- 6.3 Summary . . . 35

7 Discussion of our results **37**
- 7.1 Modifying interaction forces . . . 37
- 7.2 Parameter values . . . 38
- 7.3 Our implementation of the simulations . . . 38
- 7.4 Summary . . . 39

8 Assessment of social force models **41**
- 8.1 Strengths of social force models . . . 41
- 8.2 Weaknesses of social force models . . . 41
- 8.3 Conclusion on the assessment . . . 42

9 Summary and conclusion **43**
- 9.1 Further work . . . 43

10 References **45**

List of Figures

1 Notation for pedestrians . . . 10
2 Our interpretation of the average velocity . . . 12
3 Interaction between two pedestrians. . . . 14
4 Notation for the interaction between pedestrians and walls . . . 15
5 A wall with a kink . . . 20
6 Preventing repulsion from doorways . . . 21
7 Images of the three scenarios we simulate. . . . 31
8 Faster-is-slower effect . . . 33
9 Lane formation from [Helbing *et al.*, 2001] . . . 34
10 Lane formation in corridor simulations . . . 34
11 Freezing-by-heating effect from [Helbing *et al.*, 2002] compared with our own. . . . 35
12 Illustration of oscillating flow from [Helbing *et al.*, 2002] . . . 36
13 Oscillatory flow simulations . . . 36
14 Friction forces . . . 37
15 Problematic behaviour . . . 38
16 Leaving time for different random seeds . . . 39

List of Tables

1 Summary of main forces and base notation. . . . 11
2 Summary of components of the desired movement force . . . 12
3 Summary of components of the interaction force . . . 14
4 Summary of components for wall repulsion. . . . 15
5 Summary of the model. . . . 16
6 Model parameters . . . 32

List of Code Segments

1 Main function calling the other parts of the calculation code. . . . 24
2 Calculating the desired force. . . . 25
3 Calculating the repulsion from other pedestrians. . . . 25
4 Calculating the repulsion from the walls. . . . 26
5 Finding the wall repulsion points. . . . 27
6 Updating the pedestrian position. . . . 28

1 Introduction

When a lot of people are gathered in a confined space, it is not immediately obvious how they behave when moving around. It is also not obvious how to design buildings and other areas where many people gather to allow for optimal passage of crowds without jamming and related problems (such as inconvenience for pedestrians and even injuries). It is cumbersome, or even dangerous, to gather many people together to do experiments, especially if one wishes to simulate a panic situation (which cannot necessarily be induced artificially). It would thus be useful to be able to model a crowd of pedestrians and their movement.

A crowd of many people is a very complex system. One of the approaches to handling this complexity is by using a computer simulation of a model derived from the behaviour of individual pedestrians, and through the outcome of this simulation be able to say something meaningful about the whole system. This approach to modelling complex systems is called agent-based modelling and is used in many different and diverse subject areas, including molecular biology, economics and physics [Ceperley, 1999; Haim Levy, 2000; Huth and Wissel, 1992].

Within the field of crowd simulations, there are various ways of formulating such an agent-based model; we have chosen to focus on an approach that models pedestrian behaviour using a concept called "social forces" [Helbing and Molnár, 1995]. This type of model describes individual behaviour borrowing concepts and notation from physics, by defining virtual "forces" representing desired movement as well as the tendency to avoid obstacles, other people etc.

Since the original formulation of the social force concept, it has been revised several times (e.g. [Helbing *et al.*, 2000]). For this project we base our analysis on the version presented in [Helbing *et al.*, 2005], as this article has a more complete description of the model than som of the others. We will, however, also include other variants where necessary. Our chosen model, and others like it, describe a set of properties of crowd behaviour that has been seen in observations of actual crowds as well as replicated in computer simulations. We think it will be interesting to go into detail with this type of model and look into how they work and what they can tell us about crowds. This brings us to our problem formulation:

1.1 Problem formulation

> What are the strengths and weaknesses of social force models?
>
> How well do social force models simulate the behaviour of crowds?

Our problem is exemplary to mathematical modelling for several reasons. First of all, it is an example of applying mathematical theory to a field outside of mathematics itself. The field we are working with (crowd simulation) is relatively new, but has some established research[1]. The model itself contains a non-trivial application of mathematics and is an example of a type of model (agent-based models) used in many cases to study a complex system of interacting pedestrians by describing the individual behaviour of the pedestrians and using simulations to derive meaningful properties for the whole system. These factors combined makes our problem exemplary.

1.2 Target audience

The target audience of this report is comprised of students of mathematics on a level of education similar to our own. This means that we consider mathematical concepts and theory that is covered in

[1] See the introduction to social force models in section 2.

the bachelor courses at the mathematics department at RUC as known subject matter, and will therefore not explain these concepts in detail. Concepts that are beyond this are explained as necessary. In the description of the simulation program code, some familiarity with basic constructs of computer programming is assumed, and are thus not explained.

1.3 Approach to answering the problem formulation

To answer the problem formulation, it is first necessary to establish the concept of social force models in more detail, as well as give an overview over the work in the field. This is presented in section 2, where we also present the results from the simulations given in the literature that we will try to replicate in our simulations.

In order to better understand the model, we are going to make our own computer simulation of it. To do so, we will first describe the model thoroughly. In section 3, we will go over the different parts and parameters of the model, and point out parts that are ambiguous or missing from the article presenting the model. Based on this description, we will discuss what is needed to go from the abstract formulation of the model to a concrete numerical simulation in section 4.

Based on our analysis of the model, we will construct a computer simulation of it, modelling cases analogous to those presented in section 2. Our implementation of the simulation is discussed in section 5, including set up of parameters and initial conditions, the structure of our simulation and how we obtain results from it.

The actual results of the simulation are addressed in section 6. Here we present the outcome of our simulation, and whether or not we have been able to replicate the results we were aiming for. Based on this presentation, we discuss the results in section 7 including possible changes that might mitigate any discrepancies between our own results and the results we are trying to replicate.

Based on our results and the proposed changes to the model, we will assess social force models in general in section 8, giving an overview of the strengths and weaknesses of this modelling approach. Finally, we will give a summary of our findings and an overview of the whole project in section 9.

2 Social force models

One would think that the motion and dynamics of a human crowd would be governed by complex human decision making. However, the idea of social force models is modelling the behaviour using only a set of simple forces to describe the behaviour of the human pedestrians that comprise a crowd [Helbing and Molnár, 1995].

This works by calculating a set of social forces affecting each pedestrian in the system. These forces are not real physical forces but rather a measure of the pedestrians' motivation for acting in specific ways. However the name *social forces* is not accidental: both notation and to some extent the interpretation is borrowed from physics.

Borrowing the concept of forces from physics means that pedestrian movement is calculated from the social forces using established methods from physics, i.e. equations of motion. This means that forces can be summed to generate a resulting force for a pedestrian, and that this resulting force can be used to describe an acceleration, and thus affect the movement of the pedestrian.

However, social forces differ from physical forces in key areas. Social forces are a way to estimate a pedestrian's behaviour and tendency to move in a certain direction. As such they do not deal with the physical effects of the environment the pedestrian is in, but rather the pedestrian's perception of this environment. For example, pedestrians' attempt to avoid colliding with each other, which is modelled as a repulsive force between them. This means that the forces are not bound by the laws of physics; they can affect pedestrians over a distance, whereas a physical force between two pedestrians only occur when they are touching. Also, the forces do not obey Newton's third law of motion, since they do not represent an actual interaction between pedestrians, but only a desire to move in a certain direction. Finally, it does not make sense to talk about the results of the forces in terms of physical forces. This means that it does not make sense to talk about, e.g., pressure by directly measuring the magnitude of the forces a pedestrian is subjected to.

Throughout the report, whenever we refer to forces we mean social forces, unless stated otherwise. In the following we give an overview of which variants of social force models exist and how they differ, and of the results that have been obtained from simulations using these models.

2.1 Variants of social force models

Social force models can be found in several variants in the literature. The basic idea is the same in all of these variants, but some details differ, such as the exact forces included in the models; sometimes even in contradictory ways. In this section we outline the different variants of social force models. As mentioned in the introduction, we base our analysis and simulations on the variant presented in [Helbing *et al.*, 2005]. Therefore, the review presented here will use this variant as a basis for comparisons.

The model we are using considers all pedestrians to be circular with a given radius. This is obviously a simplification, and indeed one of the advances in later versions of social force models, is introducing elliptically shaped pedestrians, which is seen to yield results that better match experiments made in real life [Johansson *et al.*, 2007].

Another modification that is seen in other articles is the adding of additional forces, such as velocity-dependent repulsive forces, or forces that are perpendicular to the direction of the repulsion [Helbing *et al.*, 2000; Johansson *et al.*, 2007]. The latter may be used both as a means of simulating a tendency of pedestrians to avoid each other, and as a frictional force when pedestrians touch.

Finally, there are models that add an additional layer on top of the social force models, to simulate higher-level decisions such as path finding and communication between pedestrians as well as incor-

porating pedestrians that get hurt in dense crowds and fall down and turn into obstacles [Pelechano *et al.*, 2007].

The different variations in the calculations of forces do not really make a difference when looking at the results, according to [Helbing *et al.*, 2005]:

> One may, of course, take into account other details such as a velocity dependence of the forces and non circular shaped pedestrian bodies, but this does not have qualitative effects on the dynamical phenomena resulting in the simulations. In fact, most observed self-organization phenomena are quite insensitive to the specification of the interaction forces, while it may, of course, influence the quantitative results.

However, in our simulations we have been unable to replicate several of the results presented elsewhere, which contradicts the above quote. This is discussed further in sections 6 and 7.

2.2 Results obtained from the models

To have something to compare our own simulation results with, we are going to attempt to replicate some of the results obtained from other social force model simulations. In this section we present the different results and their sources. We also list the scenarios we are going to simulate to replicate the results.

There are four main results we are going to attempt to replicate: lane formation, the "freezing-by-heating effect", oscillatory flows at bottlenecks and the "faster-is-slower effect".

- **The "faster-is-slower effect"** appears when a crowd of pedestrians try to exit through a narrow passageway, such as a door leading out from a room. It has been observed that when pedestrians attempt to move faster (e.g. in panic situations) it will actually result in a *longer* time for all pedestrians to leave the room, because pedestrians block each other on the way out [Helbing *et al.*, 2000].

- **Lane Formation** occurs when pedestrians are walking towards each other from opposite directions. The effect is caused by pedestrians walking in the same direction lining up behind each other and walking in lanes. This effect is seen in simulations as well as in observations of real-life crowd behaviour and is obvious when viewing illustrations of the simulation [Helbing *et al.*, 2005].

- **The "freezing-by-heating effect"** occurs when lanes break down and clogging occurs (due to e.g. a high density of pedestrians) that makes the pedestrians come to a complete halt. It is not clear why it is called freezing by heating, but this is the term used in the article that describes it, so we refer to it by this name [Helbing *et al.*, 2005].

- **Oscillatory Flows at Bottlenecks** arise when pedestrians are moving in opposite directions and pass through a bottleneck, i.e. a narrowing of the passageway. When it is not possible for groups of pedestrians to pass each other at the bottleneck, crowds will line up on both sides of the bottleneck, trying to push through. This results in an oscillatory flow when small groups of pedestrians will pass through in one direction, making room for a small group to pass through in the other, and so on. This can be seen as oscillations in the measured flow rate in each direction and has been seen in both simulations and real life observations [Helbing *et al.*, 2005].

2.3 Simulation scenarios

To attempt to replicate the results presented in the preceding section, we will simulate two main scenarios: a square room and a corridor, with a few variations.

To replicate the faster-is-slower effect, we will simulate a square room filled with pedestrians, that all try to exit through the same exit. To replicate the other effects, we will simulate a corridor in two variations. In the simple case, we will simulate a plain corridor with pedestrians walking in both directions, and see if we get lane formation and freezing by heating. Finally, we will simulate a corridor with a narrowing in the middle to replicate the oscillatory flows at bottlenecks. The results from our simulations are presented in section 6 and discussed in section 7.

2.4 Summary

In this section we have presented the concept of social force models, and how social forces differ from physical forces. We have given an overview of the variations of social force models presented in the literature, and we have presented four results that we will attempt to replicate: the faster-is-slower effect, lane formation, freezing by heating and oscillatory flows at bottlenecks. Finally, we have presented the scenarios we will simulate to try to replicate the results.

In the next section, we will present the model we are simulating in detail.

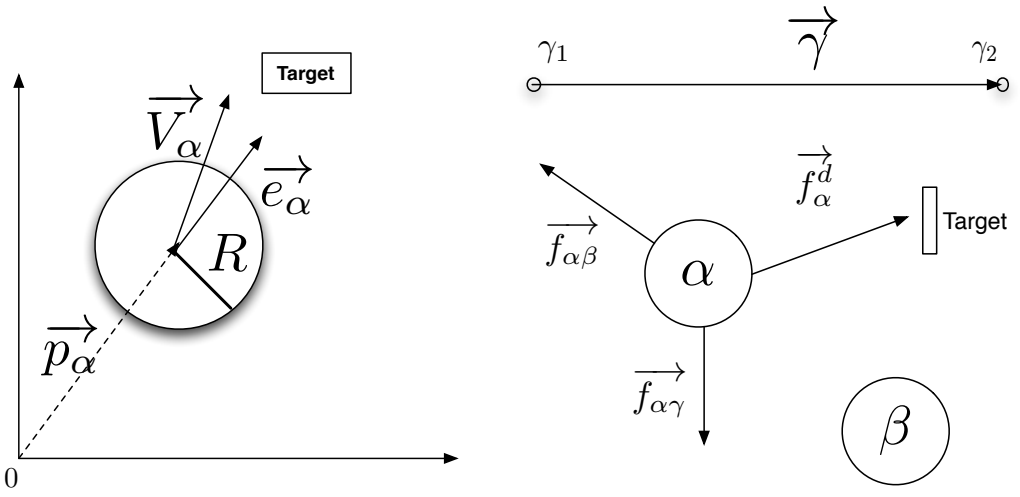
3 The model

In this section we will go through the model presented in [Helbing *et al.*, 2005] in detail. As mentioned in the introduction, social force models are agent-based, that is they describe the system by looking at the behaviour of each agent or, in our case, pedestrian in the system.

In this model the behaviour of each pedestrian is defined by a series of social forces. We will describe three main forces, that each represent a tendency of the pedestrians. These forces are *the desired movement force*, *the interaction force* between pedestrians and *the repulsive force* from the walls. The model also contains an attractive force, allowing e.g. a group of pedestrians to stick together when moving around, and a fluctuation term, adding a stochastic quality to the model. Both of these forces, however, are only mentioned in the article, and are not really used. We have written to the authors of the article asking about this, and they have replied that these parts of the model have not been used in practice, but that they might be useful to include at a later point in time. Because of this, we will exclude these parts from the explanation of the model.

In the following, we will go through the three main forces, explaining how they are calculated. Since the notation used in the article is ambiguous and confusing, we introduce our own notation in an attempt to make it clearer. We will not refer to the original notation, but only use our own.

In the model, a pedestrian is represented by a circle with a centre, a radius and a current velocity. Walls are represented as line segments. The desired direction is represented by assigning to each pedestrian a target to move towards. Various other parameters are assigned to each pedestrian; these will be explained when relevant throughout the description. The notation for pedestrians and the main forces acting on a pedestrian are illustrated in figure 1.



(a) Properties describing a pedestrian.

(b) Forces acting on a pedestrian.

Figure 1: Notation for pedestrians. (a) A pedestrian is described by a position vector ($\overrightarrow{p_\alpha}$), a velocity vector ($\overrightarrow{V_\alpha}$), a vector pointing towards the target ($\overrightarrow{e_\alpha}$) and a radius ($R_\alpha$). The velocity is determined from the forces action upon the pedestrian. (b) The forces acting on a pedestrian α is the interaction force from pedestrian β ($\overrightarrow{f_{\alpha\beta}}$), the repulsive force from wall γ ($\overrightarrow{f_{\alpha\gamma}}$) and the force towards the target ($\overrightarrow{f^d_\alpha}$).

The resulting force for pedestrian α, $\overrightarrow{f_\alpha}$ is the sum of the three main forces:

$$\overrightarrow{f_\alpha} = \overrightarrow{f_\alpha^d} + \sum_{\beta \neq \alpha} \overrightarrow{f_{\alpha\beta}} + \sum_{\gamma} \overrightarrow{f_{\alpha\gamma}} \tag{1}$$

Here $\overrightarrow{f_\alpha^d}$ is the desired movement force, $\overrightarrow{f_{\alpha\beta}}$ is the interaction force from pedestrian β and $\overrightarrow{f_{\alpha\gamma}}$ is the repulsive force from wall γ. The main forces are summarised in table 1 and will be described in detail in the following sections.

Main forces	
$\overrightarrow{f_\alpha^d}$	Desired movement force
$\overrightarrow{f_{\alpha\beta}}$	Interaction force from pedestrian β
$\overrightarrow{f_{\alpha\gamma}}$	Repulsive force from wall γ
Base notation	
$\overrightarrow{p_\alpha}$	Position vector of pedestrian α
$\overrightarrow{V_\alpha}$	Velocity of pedestrian α
R_α	Radius of pedestrian α
$\overrightarrow{e_\alpha}$	Unit vector towards α's target.

Table 1: Summary of main forces and base notation.

3.1 The desired movement force

The *desired movement force* represents the pedestrians' desire to move towards its target. The idea is that a pedestrian, if unhindered, will move towards its target at an initial desired speed that is given as a model parameter. The desired movement force is a velocity dependent force given by:

$$\overrightarrow{f_\alpha^d}(t) = \frac{1}{\tau}\left(V_\alpha^d(t)\overrightarrow{e_\alpha} - \overrightarrow{V_\alpha}(t)\right) \tag{2}$$

Here $V_\alpha^d(t)$ is the desired speed at time t, $\overrightarrow{e_\alpha}$ is the unit vector pointing towards the pedestrian's target and $\overrightarrow{V_\alpha}(t)$ is the actual velocity of the pedestrian at time t. It is not clear from the article what the initial velocity of the pedestrians is (i.e. $\overrightarrow{V_\alpha}(0)$), so we have assumed it to be zero.

τ is the *relaxation time* and scales the force, determining how fast a pedestrian changes velocity and returns to its desired velocity after having been walking slower because of obstacles etc. The relaxation time is a model parameter that in principle can vary for each pedestrian. However, from [Helbing *et al.*, 2005] it is clear that in practice, τ is the same for all pedestrians in the simulation.

$V_\alpha^d(t)$ is the desired speed of the pedestrian. The desired speed can vary over time and is given by:

$$V_\alpha^d(t) = \left(1 - \eta_\alpha(t)\right) V_\alpha^{Id} + \eta_\alpha(t) V_\alpha^{\max} \tag{3}$$

Here V_α^{Id} is the *initial desired speed* (a model parameter), and $V_\alpha^{\max}$ is the *maximum desired speed* of pedestrian α. The maximum desired speed is given as $V_\alpha^{\max} = CV_\alpha^{Id}$, where C is some constant larger than 1, given as a parameter of the model. This speed is the maximum speed the pedestrian will try to accelerate to when compensating for slowing down because of obstacles.

The value $\eta_\alpha(t)$ is called the *impatience factor* of the pedestrian and is given by:

$$\eta_\alpha(t) = 1 - \frac{\langle V_\alpha(t) \rangle}{V_\alpha^{Id}} \tag{4}$$

where $\langle V_\alpha(t) \rangle$ is the average speed in the desired direction for all times $0 \ldots t$. It is not clear from the article exactly how this average speed is calculated. We have defined it by projecting the vector pointing from the pedestrian's initial position p_α^i to the current position p_α, onto the vector pointing from p_α^i to the pedestrian's target. The length of this projection is divided by the time to yield the average speed. An illustration of this interpretation is given in figure 2.

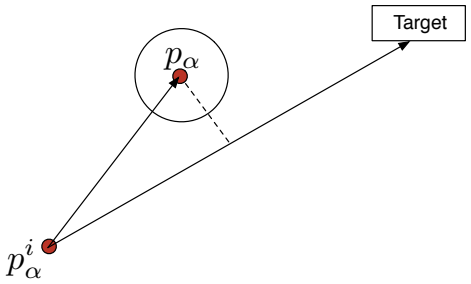


Figure 2: Our interpretation of the average velocity. The vector pointing from the pedestrian's initial position p_α^i to the current position p_α, is projected onto the vector pointing from p_α^i to the pedestrian's target. The length of this projection is divided by the time to yield the average speed towards the target.

$V_\alpha^d(t)$ is undefined at $t = 0$. We complete the definition by defining $V_\alpha^d(0) = V_\alpha^{Id}$. This means that $V_\alpha^d(t)$ is defined as follows:

$$V_\alpha^d(t) = \begin{cases} V_\alpha^{Id} & \text{if } t = 0 \\ \left(1 - \eta_\alpha(t)\right) V_\alpha^{Id} + \eta_\alpha(t) V_\alpha^{\max} & \text{if } t > 0 \end{cases} \tag{5}$$

Analysing equations (3) and (4), it is clear that as the average speed approaches V_α^{Id} so does the desired speed, since $\langle V_\alpha(t) \rangle \to V_\alpha^{Id}$ implies $\eta_\alpha(t) \to 0$ which implies $V_\alpha^d(t) \to V_\alpha^{Id}$. Correspondingly, $\langle V_\alpha(t) \rangle \to 0$ implies $\eta_\alpha(t) \to 0$ which implies $V_\alpha^d(t) \to V_\alpha^{max}$.

Since $V_\alpha^{\max}$ is set to be larger than V_α^{Id}, this means that as the average speed goes up, the desired speed goes down. When the desired speed is lower than the actual speed, this will result in a negative desired movement force due to (2), and vice versa. An interpretation of this is that the desired movement force acts as a stabilising force, that accelerates the pedestrian when it is moving slower than its desired speed, and decelerates it when it is moving faster, allowing for movement speeds of up to the maximum desired speed to "make up" for lost time if the pedestrian has been slowed down by the other forces acting on it. The components of the desired movement force are summarised in table 2.

Components of the desired movement force	
$\overrightarrow{V_\alpha}(t)$	Velocity of pedestrian α at time t
$V_\alpha^d(t)$	Desired speed of pedestrian α at time t
V_α^{Id}	Initial desired speed of pedestrian α
$\langle V_\alpha(t) \rangle$	Average speed of pedestrian α
τ	Relaxation time

Table 2: Summary of components of the desired movement force

3.2 Interaction force between pedestrians

The *interaction force* between pedestrians is a force acting between all pedestrians in the simulation. The total interaction force affecting pedestrian α is the summation of the forces between it and all other pedestrians in the simulation. The interaction force is repulsive in nature, and prevents pedestrians from overlapping or walking through each other if the magnitude of the force is sufficiently large. In the following, the force acting between two pedestrians α and β is described when calculated for α.

The variant of this force that we present is slightly different from the one in [Helbing *et al.*, 2005]. The force from the article contained two parts with different constants but otherwise identical; however, no explanation was given for how to obtain one set of constants. We have written the authors of the article, and they suggested we use the values from a newer article, [Johansson *et al.*, 2007]. The force presented in this article corresponds to collapsing the two parts given in the original article into one, and adjusting the constants accordingly.

The function for the interaction force between pedestrians depends on the position and velocity of both pedestrians, and is given by:

$$\overrightarrow{f_{\alpha\beta}}(t) = w\left(\theta_{\alpha\beta}\right)\overrightarrow{g}\left(d_{\alpha\beta}(t)\right) \tag{6}$$

Here $\theta_{\alpha\beta}$ is the angle between the movement direction of α and the vector pointing from α to β and $d_{\alpha\beta}$ is the distance between the centres of the pedestrians, as seen in figure 3a on the following page.

$w(\theta_{\alpha\beta})$ is given as:

$$w\left(\theta_{\alpha\beta}\right) = \left(\lambda + (1-\lambda)\frac{1+\cos\theta}{2}\right) \tag{7}$$

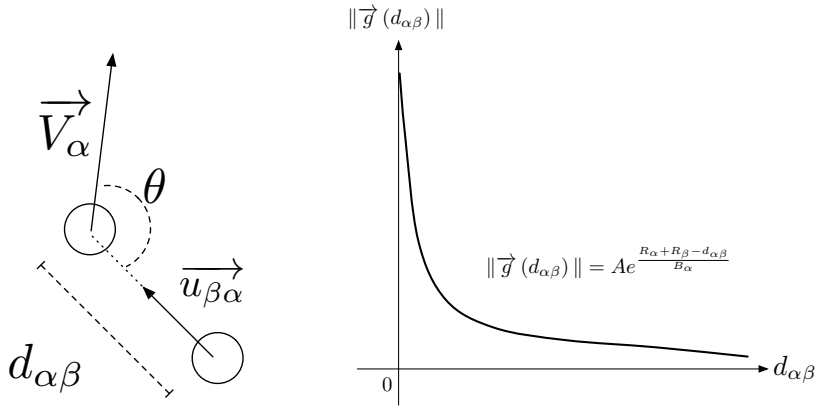
Where $0 \leq \lambda \leq 1$ is a parameter of the model. This, and the fact that $-1 \leq cos(\theta_{\alpha\beta}) \leq 1$, implies that $\lambda \leq w \leq 1$. So w modifies the force given by $\overrightarrow{g}\left(d_{\alpha\beta}(t)\right)$ by an angular component. When $\lambda = 1$, so is w, and no angular modification is made. When $\lambda < 1$, the force is modified by a weight determined by the angle, giving higher weights to angles closer to 0 and lower weights to angles closer to π. This corresponds to the pedestrian paying more attention to (and thus interacting more strongly with) other pedestrians in front of it, than behind it. The λ parameter determines how pronounced this anisotropic property of the model is.

The interaction force, $\overrightarrow{g}\left(d_{\alpha\beta}(t)\right)$, is given by:

$$\overrightarrow{g}\left(d_{\alpha\beta}\right) = Ae^{\left(\frac{R_\alpha + R_\beta - d_{\alpha\beta}}{B}\right)}\overrightarrow{u_{\beta\alpha}} \tag{8}$$

Here $\overrightarrow{u_{\beta\alpha}}$ is the unit vector pointing from β to α and A and B are model parameters. In the original article it is specified that they may vary between pedestrians. However, in practice they have a single value, so we treat them as such. The notation for the interaction force is summarised in figure 3a.

The relation between the repulsive force from β to α is seen in figure 3b. The force increases exponentially the closer α gets to β. This means that the force will hardly be noticeable for pedestrians that are far away from each other, but is strong enough when they are close to each other to prevent them from colliding, without adding any physical interaction forces. The parameter B determines how fast the force decreases with respect to the distance between the pedestrians, and with a big value of B the force decrease slowly. The strength of the force depends on the parameter A.



(a) Notation for interaction

(b) Relation between distance and force

Figure 3: Interaction between two pedestrians. (a) The interaction between two pedestrians is governed by the distance between pedestrians $d_{\alpha\beta}$, the unit vector $\overrightarrow{u_{\beta\alpha}}$ pointing from β to α, and the angle θ between α's direction of movement and the vector pointing from α to β. (b) the repulsive force away from β increases exponentially the closer α gets to β.

Components of the interaction force	
$\theta_{\alpha\beta}$	Angle between α's direction of movement and the vector from α to β
$d_{\alpha\beta}$	Distance between pedestrians α and β
$\overrightarrow{u_{\beta\alpha}}$	Unit vector pointing from β to α
λ	Parameter controlling the anisotropic property of the interaction
A	Parameter controlling the interaction strength
B	Parameter controlling the range of the repulsive interaction

Table 3: Summary of components of the interaction force

3.3 Repulsion from the walls

The *repulsive force* from the walls is the force that measures the pedestrian's desire to avoid the wall, and prevents pedestrians from walking into, or even through, walls in the simulation. As with the interaction force between pedestrians, the repulsive force from the walls functions without taking into account the physical forces between walls and pedestrians.

The repulsion from the wall γ is given by:

$$\overrightarrow{f_{\alpha\gamma}}(t) = -\nabla_{\overrightarrow{p_\alpha}} h\left(d_{\alpha\gamma}(t)\right) \tag{9}$$

Here h is a repulsive potential and $d_{\alpha\gamma}(t)$ is the distance from the pedestrian to the nearest point on the wall at time t. $\nabla_{\overrightarrow{p_\alpha}}$ is the gradient of the repulsive potential with respect to the pedestrian's position. The notation for the wall interaction is summarised in figure 4.

It is not clear from the article how finding the nearest point on the wall is accomplished. In most cases this can be done by projecting the position of the pedestrian onto the line segment that defines a wall. Some complications arise because the projection it not always a point on the wall; this is discussed further in section 4.3. In the following, we assume that the nearest point on the wall, $\overrightarrow{p_{\gamma\alpha}}$, is known.

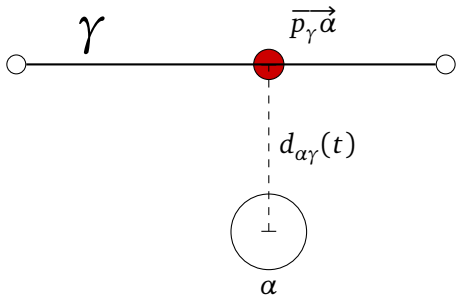


Figure 4: Notation for the interaction between pedestrians and walls. $\overrightarrow{p_{\gamma\alpha}}$ is the point on the wall γ nearest to pedestrian α and $d_{\alpha\gamma}(t)$ is the distance between the pedestrian and this point.

The definition of $h\left(d_{\alpha\gamma}(t)\right)$ is not given in [Helbing *et al.*, 2005]. However, as with the interaction force between pedestrians, it is given in [Johansson *et al.*, 2007] by:

$$h\left(d_{\alpha\gamma}(t)\right) = Ue^{\frac{-d_{\alpha\gamma}(t)}{R_\alpha}} \tag{10}$$

where U is a constant given as a model parameter.

Since $p_{\gamma\alpha}$ and p_α are known, the gradient can be calculated only for the magnitude of the force, adding the direction later. Given this interpretation, the wall interaction force becomes:

$$f_{\alpha\gamma}(t) = -\left(\frac{\partial}{\partial d_{\alpha\gamma}(t)} Ue^{\frac{-d_{\alpha\gamma}(t)}{R_\alpha}}\right)\overrightarrow{u_{\gamma\alpha}} \tag{11}$$

where $\overrightarrow{u_{\gamma\alpha}}$ is the unit vector pointing from γ to α.

Differentiating this gives:

$$f_{\alpha\gamma}(t) = \frac{U}{R_\alpha} e^{\frac{-d_{\alpha\gamma}(t)}{R_\alpha}} \overrightarrow{u_{\gamma\alpha}} \tag{12}$$

Since the distance between the pedestrian and the wall cannot be negative, the exponential term of the repulsion will always be between zero and one, rising towards one as the distance decreases. This means that $0 < |f_{\alpha\gamma}(t)| < U/R_\alpha$, increasing exponentially towards U/R_α as the distance between the pedestrian and the wall decreases. Similar to the interaction force between pedestrians, this means the repulsion from the walls is negligeble when the distance is large, but strong enough to prevent pedestrians from walking into or through walls even when no physical forces between them are present. The model parameter, U, determines exactly how strong this repulsive force is.

The parameters for the wall repulsion is summarised in table 4.

Compontents for wall repulsion	
$\overrightarrow{p_{\gamma\alpha}}$	Point on the wall γ closest to the pedestrian.
$d_{\alpha\gamma}(t)$	Distance from the wall γ to pedestrian α.
U	Constant determining the magnitude of the repulsive force from walls.

Table 4: Summary of components for wall repulsion.

3.4 Summary

As we have seen, the model consists of three main parts: the desired movement force, the interaction force between agents and the repulsive force from walls. The desired movement force represents the pedestrians desire to move with a certain velocity. The force accelerates the pedestrian to this velocity. The interaction force prevents pedestrians from walking into each other. It is a repulsive force between pedestrians that decreases exponentially with the distance between them. Model parameters determines the magnitude and range of the force. Finally, the repulsive force from the walls prevents pedestrians from bumping into and walking through walls. This force, like the interaction force, decreases exponentially with distance, and its magnitude is determined by a model parameter.

A complete summary of the model's notation and parameters is given in table 5.

Our review of the model has revealed various areas that we need to address in order to create practical numerical simulations. We have already filled out some gaps by adding details from articles other than the original, but the model as we have presented in this article is still quite abstract. In the following section, we outline the parts missing to turn the model into a simulation, and how we have solved these problems.

	Main forces
$\overrightarrow{f_\alpha^d}$	Desired movement force
$\overrightarrow{f_{\alpha\beta}}$	Interaction force from pedestrian β
$\overrightarrow{f_{\alpha\gamma}}$	Repulsive force from wall γ
	Base notation
$\overrightarrow{p_\alpha}$	Position vector of pedestrian α
$\overrightarrow{V_\alpha}$	Velocity of pedestrian α
R_α	Radius of pedestrian α
$\overrightarrow{e_\alpha}$	Unit vector towards α's target.
	Components of the desired movement force
$\overrightarrow{V_\alpha}(t)$	Velocity of pedestrian α at time t
$V_\alpha^d(t)$	Desired speed of pedestrian α at time t
V_α^{Id}	Initial desired speed of pedestrian α
$\langle V_\alpha(t)\rangle$	Average speed of pedestrian α
τ	Relaxation time
	Components of the interaction force
$\theta_{\alpha\beta}$	Angle between α's direction of movement and the vector from α to β
$d_{\alpha\beta}$	Distance between pedestrians α and β
$\overrightarrow{u_{\beta\alpha}}$	Unit vector pointing from β to α
λ	Parameter controlling the anisotropic property of the interaction
A	Parameter controlling the interaction strength
B	Parameter controlling the range of the repulsive interaction
	Components for wall repulsion
$\overrightarrow{p_{\gamma\alpha}}$	Point on the wall γ closest to the pedestrian.
$d_{\alpha\gamma}(t)$	Distance from the wall γ to pedestrian α.
U	Constant determining the magnitude of the repulsive force from walls.

Table 5: Summary of the model.

4 From model to simulation

In this section we outline the additional assumptions and methods that are needed to turn the model as described in section 3 into a numerical simulation on a computer. This includes clearing up ambiguities and filling in gaps in the model definition. The goal is, of course, to actually create a working simulation of the model. Some of the components described in this section we have come up with ourselves, and others we have collected from various other publications related to social force models. References are noted where applicable.

The questions we resolve in this section are how to approximate the movement of pedestrians, how to set initial conditions and constants and how to identify the points on the walls to calculate repulsion from. Approximation of the movement is needed when turning the model into a concrete numerical simulation. Initial conditions are needed to set up the simulation, but not all parameters are given in the articles. Finally, as it is noted in section 3.3, in some geometrical cases, there are some ambiguities that become apparent when determining which points to use in the calculation of the repulsion from the walls. These problems are discussed in the following sections.

4.1 Approximating pedestrian movement

The model can be viewed as representing $4n$ coupled differential equations, where n is the number of pedestrians. These differential equations describe the positions and speed of movement of the pedestrians in the x and y directions. Since we cannot solve these equations analytically, we will approximate them by doing numerical simulations in discrete time steps. There are various ways of approximating differential equations such as those represented by the model; we have chosen to use Euler's method as it is simple to work with. Euler's method is given by a truncated Taylor expansion [Frenkel and Smit, 2002]:

$$p(t + \Delta t) = p(t) + V(t)\Delta t + \frac{1}{2} f(t)\Delta t^2 \tag{13}$$

Here p is the position, V is the velocity, f is the total force and Δt is the time step. Similarly we have the velocity given as:

$$V(t + \Delta t) = V(t) + f(t)\Delta t \tag{14}$$

Euler's method, while simple, is not a very precise way of estimating the equations, and is very sensitive to choosing a suitable time step. Various more advanced methods of choosing the time step size as well as better algorithms to estimate the equations exist, such as the Runge-Kutta method [Frenkel and Smit, 2002; Butcher, 2003]. However, these methods are much more complicated to work with for a model as complex as ours. And from experimenting with various time step values, we have found that using Euler's method and setting a suitably small static time step provides reasonable results. Since this is the case, and since Euler's method is specifically mentioned as the method used in [Johansson *et al.*, 2007], we have chosen to stick with this method despite its disadvantages and not complicate our simulations further by using another method.

4.1.1 Choosing the time step

Choosing a suitable time step size is important when applying Euler's method. There are two main trade-offs in selecting this step size: precision of the approximation and numerical rounding errors.

If the step size is too large, Euler's method will yield an approximation that differ too much from the actual values of the equations. In our case this would result in pedestrians changing direction and velocity too slowly, making them e.g. move through a wall between two time steps. However, lowering the time step size will, apart from resulting in longer calculation times, potentially increase the error introduced by rounding errors. Rounding errors occur because numerical calculations have a finite precision and so results have to be truncated. The more calculations occur, the larger this error becomes. This means that setting the time step too low will result in too many errors from this source [Arciniega and Allen, 2003].

4.2 Initial conditions and constants

As seen in section 3, there are various parameters that must be set in order to simulate the model. Apart from this, there are also various initial conditions (such as the pedestrians' starting position) that must be set. In this section, we outline these parameters and initial conditions, describe how we have obtained their values and how we plan to vary them when we run the simulations.

4.2.1 A note on random numbers

Some of the initial conditions are set using random distributions, either uniformly or Gaussian distributed, in order to simulate natural variations in e.g. the radius of pedestrians. These random numbers are created by the pseudorandom number generator built in to the Python programming language. This generator is an implementation of the Mersenne twister generator[2] and we consider it sufficient for our purposes. This type of pseudorandom number generator is deterministic for identical seed values, which means we can run multiple simulations of the same initial conditions by fixing the seed of the random number generator to the same value for each run.

Of course the use of (pseudo)random numbers may be a source of errors. We discuss this further in section 7.3.

For the initial conditions that are Gaussian distributed, we obtain distributions by using the distribution functions of the *NumPy* mathematical library for Python [NumPy, 2010].

4.2.2 Scenario geometry

We set various parameters that describe the geometry of the scenarios we want to simulate. We have defined these ourselves, from the scenario descriptions given in section 6. These parameters are:

- **Placement of walls:** The placement of the walls is described by defining the endpoints describing the line segments that constitute each wall.
- **Pedestrian starting areas:** For each scenario we define one or more areas in which the pedestrians start. The pedestrians are distributed randomly within these areas, as described below.
- **Pedestrian targets:** Pedestrians move toward their assigned target, as described in section 3.1. Targets do not differ between pedestrians, except that we assign a target for each starting area to allow for simulating scenarios where pedestrians move towards each other in groups.

  Since the model does not deal with path finding, targets are not changed during the simulation. When a pedestrian reaches its target it is removed from the simulation.

[2] See `http://en.wikipedia.org/wiki/Mersenne_twister`

4.2.3 Pedestrians' parameters

There are a number of parameters that relate to the pedestrians. They are:

- **Pedestrians' radii:** The radii of the pedestrians, R_α, is drawn from a Gaussian distribution with a mean of 0.3 meters and a standard deviation of 0.05 meters. This is done to simulate a natural variety in the width of human shoulders, and to avoid deadlocks caused by perfectly symmetrical forces that might otherwise occur. This method of assigning pedestrian radii is taken from [Helbing *et al.*, 2005].
- **Pedestrian positions:** Pedestrians start out randomly distributed throughout the designated starting areas defined for each scenario. We have not found an explanation for how this is done in any of the articles, so we have come up with the following algorithm for placing pedestrians:

  To avoid having pedestrians start out on top of each other, the starting area is divided into a grid with cells a little larger than the size of the maximum diameter of the generated pedestrians. The pedestrians are then placed into random cells of the grid, with the limitation that each cell can only hold one pedestrian. This makes the starting configuration of the pedestrians a little more rigid than what might otherwise be observed, but we feel this is acceptable since pedestrians move out of these grid cells within the first few time steps.
- **Initial velocity:** As described in section 3.1, the initial pedestrian velocity is assumed to be zero.
- **Initial desired speed:** The initial desired speed, V_α^{Id}, for the pedestrians is set to be Gaussian distributed with a mean of 1.3 and a standard deviation of 0.3. These values are taken from [Helbing *et al.*, 2005].
- **Maximum desired speed :** The maximum desired speed is set to $V_\alpha^{max} = 1.3V_\alpha^{Id}$. As the initial desired speed this value is also taken from [Helbing and Molnár, 1995].
- **Relaxation time:** The relaxation time, τ, is set to 1 second in [Helbing *et al.*, 2005].

4.2.4 Constants

The model includes a number of constants. These are parameters that do not vary between the pedestrians, but are fixed for the whole simulation. They are:

- **Time step size:** We have experimented with various time step sizes, and have found that with a fixed value of 0.01 seconds, we get reasonable run times and results from our simulation.
- **A, B:** We set $A = 3.0$ and $B = 0.2$, taken from [Helbing *et al.*, 2005].
- **U:** Is not given in [Helbing *et al.*, 2005], so instead we have taken it from [Helbing and Molnár, 1995] where it is set to be 10.
- **λ:** We set $\lambda = 0.75$, from [Helbing *et al.*, 2005].

4.2.5 Varying the parameters

A way of analysing the behaviour of the model would be systematically varying different parameters to assess how this affects the model behaviour and to find out which parameter values makes the model break down. However, since we have several parameters that could be varied (10 in total, not counting the scenario geometry), it is impractical to do this manually. Doing it automatically would require us to develop measures of simulation viability that can be measured automatically, and since our assessment

of which simulations are valid very much relies on visual inspection (e.g. if pedestrians walk through walls then the simulation is considered invalid), we would have to come up with an exhaustive set of tests that would tell us whether or not simulations are valid. Our cursory testing of arbitrarily chosen parameter values indicates that there are several ways in which simulations can be invalid, so we have deemed them out of scope of this project to develop such tests, and we have chosen instead to abandon the attempt to do systematic parameter variations.

Instead, we have started running our simulations using the parameter values outlined above, and then only vary parameters when necessary to get useful simulation results. While this does not give us an exhaustive overview of the model behaviour, it is sufficient to test whether or not we see the phenomena we are trying to reproduce.

4.3 Finding the nearest point on a wall

As mentioned in section 3.3, it is not clear from the article how the nearest point $p_{\gamma\alpha}$ on a wall from a given pedestrian is found. In many cases it is obvious: $p_{\gamma\alpha}$ is the point of intersection of the wall and a line perpendicular to the wall passing through p_α (the projection of p_α onto the wall).

However, since walls are described as line segments, such a point is not guaranteed to exist; the pedestrian may be positioned off either end of the wall. In the simple case, this is easily remedied: simply define $p_{\gamma\alpha}$ as the projection of p_α onto the wall, and if no such projection exists, define $p_{\gamma\alpha}$ to be the wall endpoint closest to p_α. This approach works except for two cases: When walls have kinks and at doorways.

4.3.1 Walls with kinks

Since each wall is defined as a line segment, a single wall cannot have kinks in it. To define such a wall, multiple line segments are defined with shared end points. When the angle between such two walls is greater than 180°, as indicated by the angle θ in figure 5, a problem appears.

The reason for this problem is that the same endpoint may be indicated for repulsion from multiple walls. Looking at figure 5, a pedestrian in area A will be repulsed both by a point on the wall γ_1, but also from the nearest endpoint of γ_2, which is shared with γ_1. A pedestrian in area B will be repulsed from both walls at the endpoint shared between γ_1 and γ_2. Both these situations will result in an extra repulsion that is the result of the way walls are represented (as line segments), and not a part of the model. To avoid this, the algorithm in 4.3.3 for selecting which points to calculate wall repulsion from has to take this into account.

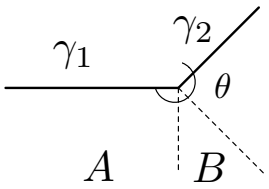


Figure 5: A wall with a kink. A pedestrian α in the area A will be repulsed by a point on the wall γ_1 and by the nearest endpoint of γ_2. In the area B, α will be repulsed twice by the endpoint shared between γ_1 and γ_2. This situation occurs when the angle $\theta > 180°$.

4.3.2 Doorways

Another problem occurs at doorways: when the wall repulsion points are the nearest endpoints of wall segments, a pedestrian passing through a doorway would be repulsed by both sides of this doorway, creating an unwanted barrier for passing through it. In simulations, this appears as a force that pedestrians have to overcome to escape through doorways.

To solve this problem, the algorithm for selecting which points on the walls to calculate the repulsion from must take into account that once a pedestrian is passing through a doorway, no repulsion should be calculated from the wall segments that form the door. A way to accomplish this is to distinguish between endpoints that are shared with other walls, and endpoints that are "free-floating". Such endpoints are only considered for repulsion if they are closer to the pedestrian than its radius, i.e. if the pedestrian touches the wall. This is illustrated in figure 6.

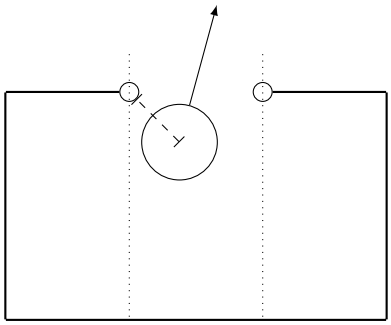


Figure 6: Preventing repulsion from doorways. To allow free passage for pedestrians between the dotted lines. This means that the pedestrian do not get repulsed from the marked endpoints unless the distance (indicated by the dashed line) is smaller than the pedestrian's radius.

4.3.3 Algorithm for identifying wall repulsion points

Based on the description above, we have devised the following algorithm for selecting which points on the walls should be included when calculating the repulsion force from walls.

The algorithm is run for each pedestrian. Its input is the pedestrian positions, p_α, and the set of starting points and endpoints of the walls. The output is the set of points from which repulsion should be calculated (one for each wall). A set of endpoints that are already used and endpoints that are candidates for repulsion are used during the operation of the algorithm.

1. For each wall, calculate the projection of the vector pointing from the wall's starting point to the pedestrian, unto the vector pointing from the wall's starting point to its endpoint.

    a) If this point is part of the wall, add it to the set of points repulsion should be calculated from, and add the wall's two endpoints to the set of already used endpoints.

    b) If the projected point is not part of the wall, instead the wall endpoint closest to the pedestrian is added to the set endpoints that are candidates for repulsion.

2. After having gone through all walls, for each point in the set of candidate endpoints, check if this point is already in the set of used endpoints. If it is, discard it.

3. Otherwise, check if the endpoint is shared with another wall (i.e. if it appears twice in the set of candidate endpoints).

4. If the endpoint is either shared with another wall or closer to the pedestrian than the pedestrian's radius, add it to the set of repulsion points and the set of used endpoints.

Our implementation of this algorithm is described in section 5.3.

4.4 Summary

In this section we have discussed issues related to turning the model description into a concrete numerical simulation. These issues include approximating the movement of pedestrians through Euler's method, setting initial constants and parameters, and an algorithm for identifying the points on the walls nearest to a pedestrian. Some of the solutions to the problems associated with turning the model into a simulation we have found solutions for in the literature, and some we have come up with ourselves. Euler's method and some of the constants belong in the former group, while other constants and the algorithm for identifying wall points belong in the latter.

In the following section, we outline how we have implemented our simulation.

5 Simulation approach

In this section, we describe how our simulation is implemented, and how the implementation works. This is done in part to document our implementation, and in part to give the reader an overview of how our results are obtained. We will describe briefly how the program is structured on a macro level, and go into more detail on the parts specific to the calculations of the model parameters. The full source code is available online[3].

Our simulation is implemented in the Python programming language, with the calculation intensive parts implemented in C for performance reasons. We assume a virtual coordinate system using meters as a base unit, and with the origin in the centre of the area we simulate. Each pedestrian is described by a centre point and a radius, and each wall is described by a line segment connecting two points.

All parameters are stored as double precision floating point values where nothing else is indicated. We use custom data structures to keep track of the pedestrians and walls while running the simulation. Python is used to set up the initial conditions, run the program's main control loop, and draw the simulation results through the *PyGame* library [PyGame, 2010]. This allows to do real-time animation as well as saving each simulation step to be assembled into a film afterwards. All calculations and data processing is done in a Python extension written in C, to increase performance. Gathering of data to draw the graphs and the drawing itself is done in the Python code.

5.1 Structure of the program

The program is structured into four main parts: The simulation calculations, drawing of the simulations, plotting of parameters and the control part setting up parameters and calling the other parts as necessary. The drawing part mainly consists of a frontend to the drawing library, and so is not interesting to discuss here. The other parts will be described in the following, structured so as to present the sequence that is followed when a simulation is run. This description consists of three parts: setting up the simulation, calculations, and gathering of results.

5.2 Setting up the simulation

The program supports defining multiple *scenarios* to simulate. Each scenario defines its own set of parameters, and a simulation run features one scenario. The parameters defined for each scenario include:

- The model constants, A, B, U, λ and pedestrian relaxation time. These are the same for all pedestrians in a simulation.
- Mean initial desired velocity and radius of pedestrians and their standard deviation, and the factor used to calculate the maximum velocity.
- The initial number of pedestrians, the area(s) they start in and the target(s) they move towards.
- The geometry of the scenario (i.e. placement of walls).
- Definition of the areas where measurement of data for plotting graphs is done and which parameters should be plotted (see section 5.4).
- Various parameters related to drawing of the simulation, maximum run time and optional continuous inflow of pedestrians.

Parameters that are not set for each scenario, but are defined once for all simulations are:

- Time step size.

[3] See `http://akira.ruc.dk/~tohojo/crowd-modelling`.

- Plotting data sampling frequency.

How the values of the parameters are determined is described in section 4.2. From these parameters, the simulation is set up by initialising the calculation module and creating the pedestrians.

The pedestrians are created with an initial velocity of zero, and distributed randomly within the area(s) designated by the parameters for the scenario, as described in section 4.2.3.

5.3 Calculation of the model

For each step of the simulation, the calculations are run in two parts: Finding the accelerations (or resulting force) for all pedestrians, and updating position and velocity for the pedestrians. Since the acceleration for each pedestrian is dependent on both position and velocity of the other pedestrians, splitting the calculations this way enables us to do the calculations of pedestrians in any order, and even parallel. The drawback is that the pedestrians are only affected by the movement and positions of other pedestrians as they were at the end of the last simulation step. This means that the time step has to be chosen appropriately, as explained in section 4.1.

The acceleration vectors for each pedestrian, α, is calculated in three steps, corresponding to the parts $\overrightarrow{f_\alpha^d}$, $\overrightarrow{f_{\alpha\beta}}$ and $\overrightarrow{f_{\alpha\gamma}}$ from section 3. In each simulation step the three forces are calculated in order, first calculating the desired force, then the repulsive force from each of the other pedestrians in the simulation, and finally the repulsive force from each wall. The resulting vectors are added to yield the total acceleration for each pedestrian.

After this is calculated for all pedestrians, their velocities and positions are updated in separate steps. Each of these four steps are described in detail in the following. The code calling the other parts of the calculation can be seen in code segment 1. This function is called once for each pedestrian (index `i`) for each simulation step.

Code segment 1: Main function calling the other parts of the calculation code.

```
1  void calculate_forces(Py_ssize_t i)
   {
       int j;
       add_desired_acceleration(&pedestrians[i]);

6      for(j = 0; j < a_count; j++) {
           if(i == j) continue;
           add_repulsion(&pedestrians[i], &pedestrians[j]);
       }

11     add_wall_repulsion(&pedestrians[i]);
   }
```

5.3.1 Calculating the desired force

The function for calculating the desired force can be seen in code segment 2. Of note is the implementation on lines 7-19 of the conditional definition of $V_\alpha^d(t)$ given in equation (5) and the creation of a unit vector in lines 20-21 to add the direction to the desired velocity.

To increase performance, it is preferred to update values in place, instead of copying them around in different parts of the computer's memory. This gives rise to the constructs around the `vector_i*`-functions in the code; here the *i* stands for "in place".

Code segment 2: Calculating the desired force.

```
static void add_desired_acceleration(Pedestrian * a)
{
    double average_velocity = 0.0, impatience = 0.0,
           desired_velocity = 0.0;
    Vector desired_direction = {0.0, 0.0};

    if(a->time) {
        double proj = vector_projection_length(
                a->initial_position, a->target, a->position);
        average_velocity = proj / a->time;

        impatience = 1.0 - average_velocity / a->initial_desired_velocity;

        desired_velocity = (1.0-impatience) * a->initial_desired_velocity + \
                           impatience * a->max_velocity;

    } else {
        desired_velocity = a->initial_desired_velocity;
    }
    desired_direction = vector_sub(a->target, a->position);
    vector_unitise(&desired_direction);

    a->acceleration = vector_mul(desired_direction, desired_velocity);
    vector_isub(&a->acceleration, &a->velocity);
    vector_imul(&a->acceleration, 1.0/a->relax_time);
}
```

5.3.2 Repulsion from other pedestrians

The two functions used for calculating the repulsion from other pedestrians is seen in code segment 3. The `add_repulsion` function is called for each pair of pedestrians. The force without any angle dependence is calculated in the `calculate_repulsion` function, and is modified by the angular dependence if the pedestrian's velocity is not zero, and λ is not one. If the velocity is zero, calculating the dot product would result in a division by zero, and if λ is one, modifying the force by the angular dependence would have no effect. The whole calculation is skipped if one of the constants A or B are zero, as this would make the calculations have no effect, or result in a division by zero, respectively.

Code segment 3: Calculating the repulsion from other pedestrians.

```
Vector calculate_repulsion(Pedestrian * a, Pedestrian * b, double A, double B)
{
    double radius_sum = a->radius + b->radius;
    Vector from_b     = vector_sub(a->position, b->position);
    double distance   = vector_length(from_b);

    vector_unitise(&from_b);
    vector_imul(&from_b, A * exp((radius_sum-distance)/B));

    return from_b;
}
```

```
void add_repulsion(Pedestrian * a, Pedestrian * b)
{
    if(!A || !B) return;
    Vector repulsion = calculate_repulsion(a, b, A, B);
    if(a->velocity.x && a->velocity.y && lambda < 1.0) {
        Vector from_a = vector_sub(b->position, a->position);

        double cosine = vector_dot(a->velocity, from_a)/(
                vector_length(a->velocity) * vector_length(from_a));

        vector_imul(&repulsion, (lambda + (1-lambda)*((1+cosine)/2)));
    }
    vector_iadd(&a->acceleration, &repulsion);
}
```

5.3.3 Repulsion from walls

The code for adding the repulsion from the walls is shown in code segment 4. The `add_wall_repulsion` function is called for each pedestrian and consists of two parts: Finding the points on the walls where repulsion should be calculated from, and calculating the repulsion from these points. The force from the repulsion points is calculated in the function `calculate_wall_repulsion` that is called for each repulsion point. This calculation corresponds to equation (12).

Code segment 4: Calculating the repulsion from the walls.

```
void add_wall_repulsion(Pedestrian * a)
{
    int i;
    Vector * repulsion_points  = PyMem_Malloc(w_count * sizeof(Vector));
    int rep_p_c = 0;
    Vector repulsion;

    rep_p_c = find_repulsion_points(a, repulsion_points);

    for(i = 0; i < rep_p_c; i++) {
        repulsion = calculate_wall_repulsion(a, repulsion_points[i]);
        vector_iadd(&a->acceleration, &repulsion);
    }


    PyMem_Free(repulsion_points);
}

Vector calculate_wall_repulsion(Pedestrian * a, Vector repulsion_point)
{
    Vector repulsion_vector = vector_sub(a->position, repulsion_point);
    double repulsion_length = vector_length(repulsion_vector);
    vector_unitise_c(&repulsion_vector, repulsion_length);

    double repulsion_force = (1/a->radius) * U * exp(-repulsion_length/a->radius);
```

```
    vector_imul(&repulsion_vector, repulsion_force);

    return repulsion_vector;
}
```

The function for finding the repulsion points on the walls is seen in code segment 5. This is an implementation of the algorithm described in section 4.3. Lines 9-27 corresponds to step one in the algorithm description that identifies definite repulsion points (those that are not wall endpoints) and possible repulsion points (those that are endpoints).

In lines 29-35 an endpoint is discarded if it is already used because it is shared with a wall that has already defined a repulsion point. In lines 36-57 endpoints that are not discarded in this way are added to the list of repulsion points if they are either not free-floating (i.e. shared with another wall), or closer to the pedestrian centre than the pedestrian's radius. This ensures that doorways do not repulse pedestrians that are passing through it.

Code segment 5: Finding the wall repulsion points.

```
int find_repulsion_points(Pedestrian * a, Vector repulsion_points[])
{
    int i,j;
    double projection_length;
    Vector * used_endpoints     = PyMem_Malloc(2*w_count * sizeof(Vector));
    Vector * possible_endpoints = PyMem_Malloc(w_count * sizeof(Vector));
    int rep_p_c = 0, use_e_c = 0, pos_e_c = 0;

    for(i = 0; i < w_count; i++) {
        Wall w = walls[i];
        projection_length = vector_projection_length(w.start, w.end, a->position);
        if(projection_length < 0)  {
            possible_endpoints[pos_e_c++] = w.start;
        } else if(projection_length > w.length) {
            possible_endpoints[pos_e_c++] = w.end;
        } else {
            // We have the length, L, of how far along AB the projection point is.
            // To turn this into a point, we multiply AB with L/|AB| and add
            // this vector to the starting point A.
            // P = A + AB*L/|AB|
            repulsion_points[rep_p_c++] = vector_add(w.start,
                    vector_mul(vector_sub(w.end, w.start),
                        projection_length/w.length));
            used_endpoints[use_e_c++] = w.start;
            used_endpoints[use_e_c++] = w.end;
        }
    }

    for(i = 0; i < pos_e_c; i++) {
        int use_e = 1;
        for(j = 0; j < use_e_c; j++) {
            if(vector_equals(possible_endpoints[i], used_endpoints[j])) {
                use_e = 0;
            }
```

```
        }
        if(use_e) {
            // Keep track of whether the endpoint is free—floating, i.e. if
            // it is shared with another wall.
            int free_e = 1;
            for(j = 0; j < pos_e_c; j++) {
                if(i != j &&
                        vector_equals(possible_endpoints[i],
                            possible_endpoints[j])) {
                    free_e = 0;
                }
            }
            // Endpoints that are free—floating (i.e. sides of doorways) are
            // only considered for repulsion if they are closer to the pedestrian
            // than the pedestrian's radius. This allows pedestrians to pass more
            // freely through doorways.
            if(!free_e ||
                    vector_length(vector_sub(a->position,
                            possible_endpoints[i])) < a->radius) {
                repulsion_points[rep_p_c++] = possible_endpoints[i];
                used_endpoints[use_e_c++] = possible_endpoints[i];
            }
        }
    }

    PyMem_Free(used_endpoints);
    PyMem_Free(possible_endpoints);

    return rep_p_c;
}
```

5.3.4 Updating position and velocity

After every pedestrian has been updated with a resulting acceleration vector from the current simulation step, all pedestrians update their position and velocity. The position is updated by calculating a displacement vector as explained in section 4.1.

After updating the position, the pedestrian's velocity is updated by adding the acceleration vector, to be used for the next simulation step. The code that does this is seen in code segment 6. Line 14 is related to the measuring of data for plotting, and will be explained in section 5.4.

Code segment 6: Updating the pedestrian position.

```
1  void update_position(Pedestrian * a)
   {
       Vector delta_p = vector_add(
               vector_mul(a->velocity, timestep),
               vector_mul(a->acceleration, 0.5 * pow(timestep, 2)));
6
       vector_iadd(&a->position, &delta_p);
       a->velocity = vector_add(a->velocity,
               vector_mul(a->acceleration, timestep));
```

```
    a->time += timestep;
11
    check_flowlines(a);
}
```

5.4 Obtaining results

There are two ways we obtain results from the simulation: through the drawings of the simulations themselves, and through graphs of various measurements made during the simulation. In this section we describe the mechanisms for obtaining these results. Which results are used for which scenarios are described, along with the scenarios themselves, in section 6.

5.4.1 Drawings of the simulation

When the simulation is run, drawings of the pedestrian's positions, walls etc. are created for each simulation step. These can be shown in real time, or saved and either assembled into a film or used as individual illustrations. We use these images to make qualitative assessments of pedestrian behaviour, e.g. determining whether a lane formation occurs. We include the relevant images as illustrations in the results section.

5.4.2 Graphs of measurements

As part of the simulation program, we have added measurements of various properties of the simulation, which are then assembled into graphs after the simulation has run. This section describes the measuring features we have added to the program, and *how* they work. *Why* we measure each property is explained along with the results. Since we have revised which graphs are needed throughout our work with the simulations, some additional measuring and graphing features exist in the source code, but are not described here.

The graphing functionality works in two different modes: One is graphing properties as a function of the simulation time, and others run multiple simulations varying one or more parameters and graphing properties as a function of this varied parameter. Any numeric parameter can be varied in this way. It is noted for each feature in which mode it is used and some properties are used for both modes. The properties we measure are:

- **Flow rate:** Flow rate is measured along a line segment defined as a parameter to the scenario; the line segment is assumed to be perpendicular to the primary movement direction of the pedestrians. The simulation keeps track of at what time each pedestrian first comes into contact with this line segment, i.e. is first less than their radius away from it . This is used as an approximation of when the pedestrians cross the line. The average flow rate is then computed as the number of pedestrians who have crossed the line divided by the time. This is graphed as a function of time, using a five second moving average, and as a function of parameters, using the average of the flow rate measurements of an entire simulation.
- **Leaving time:** The leaving time is a measure of how long it takes for all pedestrians to leave the simulation completely. Pedestrians are removed from the simulation when they reach their target, or when they get outside the calculation area which is the basis of this measurement. This property is only measured for a whole simulation, and as such is only graphed as a function of the varying parameters. Of course simulations that do not have a finite running time (i.e. where

pedestrians are added continuously, or where they never reach their target) do not measure this property.

5.5 Summary

In this section we have described our implementation of the simulation. The program is split up into three parts: setting up the simulation with parameters and initial conditions, the calculations of the model itself, and measurements of the results. The first and last parts are implemented in the Python programming language, while the calculations themselves are implemented in C for performance reasons. The measurements of simulation results include drawing of pictures of the simulation, and graphs of the flow rate and the time it takes all pedestrians to leave a room.

6 Results

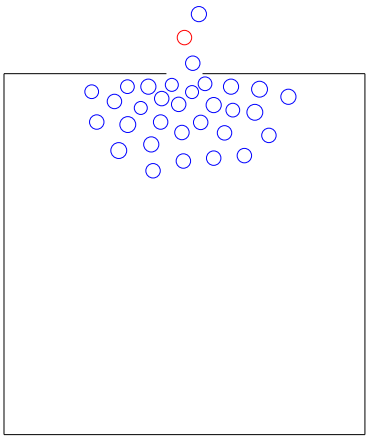
In this section we describe the results obtained from our simulations. The section is divided into two parts. In the first part we present some general results of our simulations, reviewing the general behaviour of our simulations. In the second part we present the results of our attempts to replicate the phenomena outlined in section 2.2. A discussion of these results is presented in section 7.

6.1 The basics

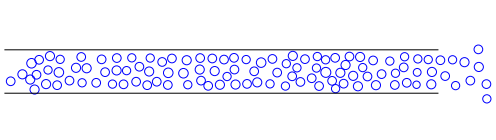
In this section we describe some general results of our simulations. First we present the scenarios that we have simulated, and the parameters used in the simulations. We then move on to comment on the general behaviour of the pedestrians in our simulations and within which limits the simulations give reasonable results.

6.1.1 Scenarios

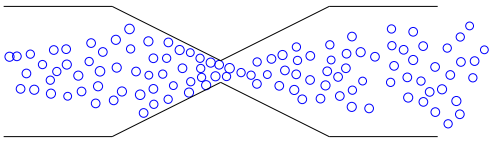
As outlined in section 2.3, we will simulate two different main scenarios, a square room and a corridor, with some variations of the latter. Counting the variations, we simulate three different scenarios: A rectangle shaped room with a door in the middle of one of the walls, and a normal corridor and a corridor with a bottleneck in the middle. In the corridors we do simulations of both uni– and birectional flow. Images of each of these scenarios are presented in figure 7.



(a) Square room.



(b) Corridor.



(c) Bottleneck.

Figure 7: Images of the three scenarios we simulate.

6.1.2 Parameters

As explained in section 4.2.5, we start out with the model parameters outlined in section 4.2 and vary them when necessary to get useful results from the simulations. The values we use initially are summarised in table 6. Whenever a parameter is changed from these values, we will point it out and explain why the change has been necessary.

Description	Symbol	Value
Constants		
Interaction strength	A	3.0
Interaction range	B	0.2
Anisotropic constant	λ	0.75
Wall repulsion strength	U	10.0
Initial conditions		
Mean pedestrian radius	R_α	0.3 ± 0.05
Mean desired velocity	V_α^{Id}	1.34 ± 0.26
Max velocity factor		1.3
Relaxation time	τ	1.0

Table 6: Model parameters used for the simulations. $\pm$ indicates standard deviation in Gaussian distributed values.

6.1.3 General behaviour of the model

Our simulations have shown that it is indeed possible to create simulations that upon visual inspection show reasonable behaviour of the pedestrians. That is, pedestrians do not pass through the walls or each other, and we see clogging of exits as we expected. However, the behaviour is very much dependent on the parameter values. This means that if, for example, the mean initial desired velocity for the pedestrians is increased, pedestrians will start walking through each other and the walls. This can be remedied by also increasing the parameters controlling the magnitude of the repulsive forces, which indicates that there is some interdependence of the parameters. We have not done a detailed analysis of this, though.

For examples of simulations we've made films available along with the source code for the simulations[4].

6.2 The results from the literature

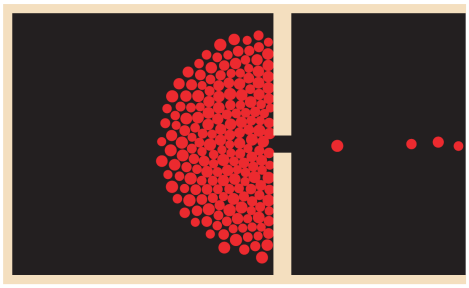
In this section we present the results we obtained from our attempts to replicate the results from the literature outlined in section 2.2. For each of the results we will go into more detail on the phenomenon we seek to replicate, and then present how well our results match the expected. The results we replicate are presented in detail in several different articles, but according to [Helbing *et al.*, 2005], they should all be possible to see using the model we are simulating.

6.2.1 The faster-is-slower effect

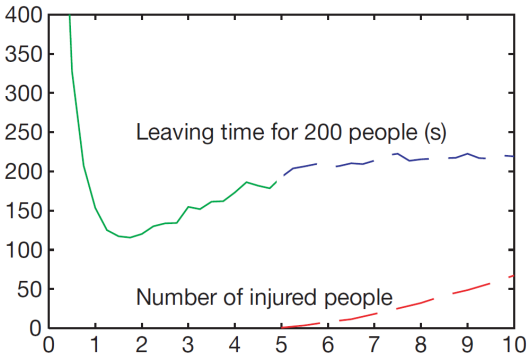
As mentioned in section 2.2, the faster-is-slower effect is seen when pedestrians leave a room. The effect is that faster desired movement speeds yield a longer time to empty the room, contrary to what

[4] See `http://akira.ruc.dk/~tohojo/crowd-modelling`.

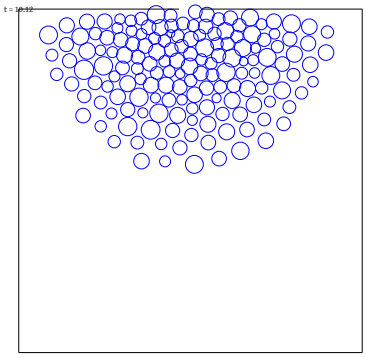
might be expected. The results from [Helbing *et al.*, 2000] where this effect is described can be seen in figure 8. Here it is apparent that clogging occurs at the doorway (figure 8a), leading to a longer time to clear the room for desired velocities above $1.5m/s$ (figure 8b).



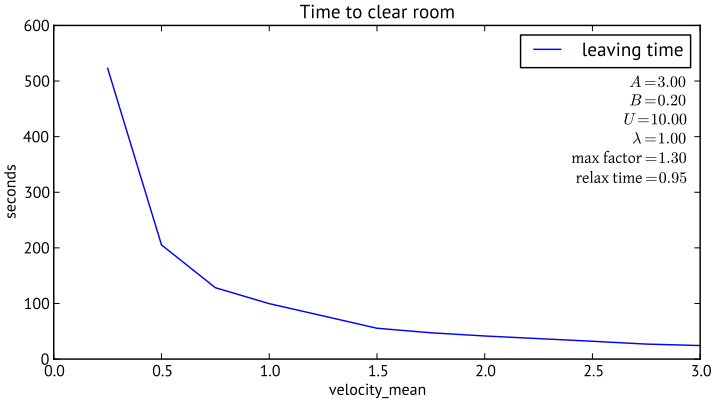
(a) Clogging at the exit.



(b) Leaving time versus desired velocity.



(c) Image from our simulations simulation



(d) Time to clear the room as a function of mean desired velocity.

Figure 8: Faster-is-slower effect. (a) and (b) from [Helbing *et al.*, 2000], (c) and (d) the corresponding data from our simulation.

In our simulations of the square room scenario we have clogging in the doorways as expected (see figure 7a).

In [Helbing *et al.*, 2000] they simulate a square room with dimensions $15\text{m} \cdot 15\text{m}$ with 200 pedestrians as shown in figure 8. In order to compare our results with theirs we simulate a corresponding scenario.

It has been necessary to lower the relaxation time of the pedestrians, τ, from 1 to 0.95 when doing these simulation because with $\tau = 1$ the pedestrians would end up in a locked state in front of the door when there are three or four pedestrians left, where the repulsion between the pedestrians prevent either of them of passing through the doorway. With a lower value of τ this effect does not occur.

It has also been necessary to remove the variation of the mean velocity when working at low velocities because some pedestrians ended up with an extremely low initial desired velocity which resulted in extremely high leaving times (for further discussion of this, see section 7.3).

When we plot the time to leave the room as a function of the initial desired velocity with values from 0.75 to 3.0, as shown in figure 8d, we do not see the leaving time increase with initial desired speed

within any specific interval. Looking at the results, we conclude that we do not see the faster-is-slower effect.

We choose to stop the initial desired speed at 3.0m/s because if you increase it beyond this value people will start going through the walls and each other. One could avoid this by also increasing the wall potential and the strength of the social force, but such adjustments are not mentioned in the article. How they have managed to run model simulations with the initial desired velocity set to 10.0m/s without seeing these errors is not clear.

6.2.2 Lane formation

Lane formation happens when the simulation contains bidirectional flow in corridors. As stated in [Helbing *et al.*, 2005]:

> If pedestrians crowds moving in opposite directions meet each other, they form small *channels* in the beginning, but these channels later merge to produce wide lanes

An illustration of this phenomenon from [Helbing *et al.*, 2001] can be seen in figure 9.

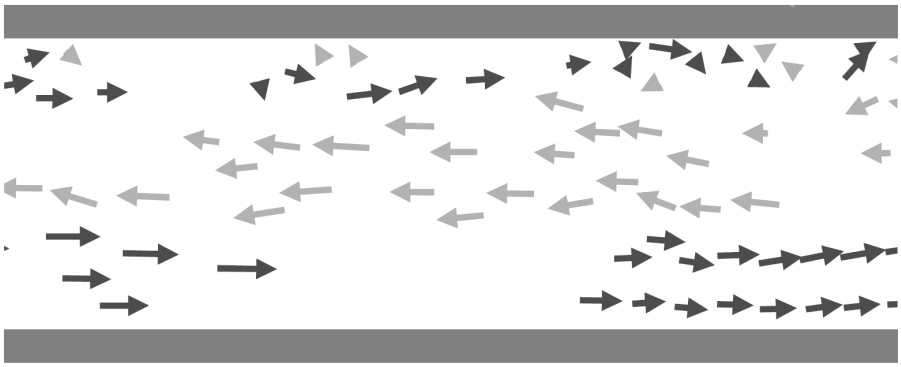
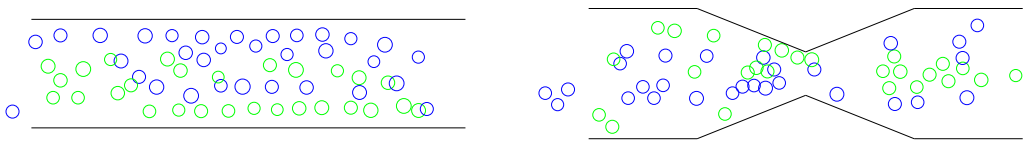


Figure 9: Lane formation from [Helbing *et al.*, 2001]. Pedestrians through a corridor form alternating lanes of pedestrians going in each direction.

Since we have two different corridor cases we have looked for lane formation in both of them. As can be seen in figure 10, the lane formation occurs in both cases.



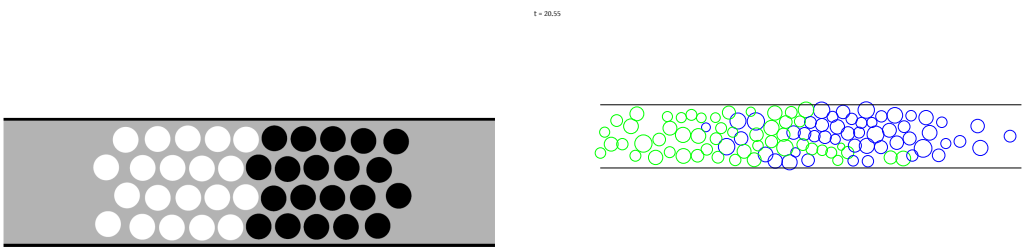
(a) Bidirectional flow in a corridor.

(b) Bidirectional flow in a corridor with a bottleneck.

Figure 10: Lane formation in corridor simulations. The colours indicate the direction the pedestrians are moving in. The blue ones are walking to the left; the green ones are walking to the right. Lane formation can be seen in both cases.

6.2.3 Freezing by heating effect

The freezing by heating effect occurs when the lane formations break down due to extreme densities. This results in a clogging that brings the pedestrians to a complete halt (see section 2.2). The illustration of this effect as presented in [Helbing *et al.*, 2002] can be seen in figure 11.



(a) Freezing by heating effect from [Helbing *et al.*, 2002].

(b) Freezing-by-heating effect from our simulation.

Figure 11: Freezing-by-heating effect from [Helbing *et al.*, 2002] compared with our own.

We have been able to reproduce the freezing-by-heating effect in both corridor cases. Actually, in most cases run with a high density of pedestrians, we see the effect in our simulations.

6.2.4 Oscillating flows

Oscillating flows occur when clogging occurs at a bottleneck, and small groups of pedestrians from each direction alternate in getting through the bottleneck. The effect is described in [Helbing *et al.*, 2002], but it is not clear how they measure it. Figure 12 on the next page shows the illustration from the article that explains the effect.

When looking at the simulation, it is apparent that clogging appears in both ends of the bottleneck and people are sometimes squeezed through the bottleneck, but it is not immediately clear whether this constitutes an oscillating flow. To get a better indication of whether or not oscillation occurs, we have graphed the flow rate in each direction. This graph is seen in figure 13. The graph does not show a clear indication of oscillating flows, however a possible source of error is that the flow rate is measured at the ends of the corridor, which may dilute the effect as pedestrians are passing through the crowd after having passed through the bottleneck. However, in the way our simulation is implemented, it is not possible to distinguish between pedestrians moving in different directions when measuring the flow rate in the middle of the bottleneck.

In conclusion, we may be able to replicate the oscillating flows, but the results are not completely clear.

6.3 Summary

We have seen that our simulation is able to show reasonable results of pedestrians moving through our chosen scenarios. We have simulated three different scenarios, and tried to replicate the results presented in the literature. We have not been able to replicate the faster-is-slower effect. Lane formation and freezing-by-heating we do see, and it is unclear whether oscillating flows are seen, perhaps partly because our way of measuring this is not very good.

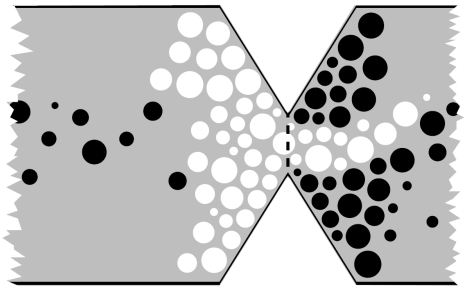
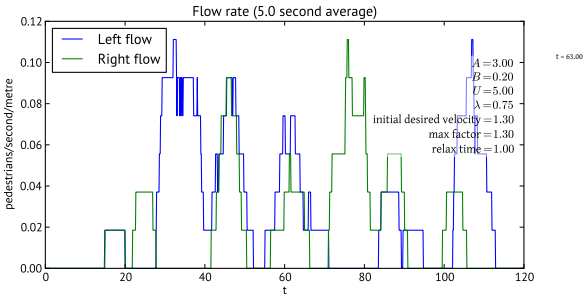
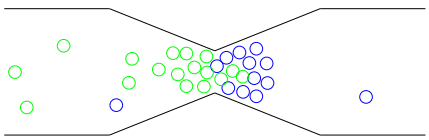


Figure 12: Illustration of oscillating flow from [Helbing *et al.*, 2002]. A small group of pedestrians slipping through the bottleneck passes through the bottleneck in one direction. Oscillations occur when pedestrians alternately pass through the bottleneck from each direction.



(a) Flowrates measured at each end of the corridor



(b) Clogging of a bottleneck

Figure 13: Oscillatory flow simulations. (a) Flowrates measured at each end of the corridor of a simulation. While there is some alternating behaviour it is not clear whether oscillatory flow occurs. (b) Clogging at a bottleneck when pedestrians approach it from both directions.

7 Discussion of our results

As we outline in section 6, we are not able to reproduce all the results from the literature in our own simulations. Through our study of social force models, we have come across a wide variety of different improvements. We believe that some of these improvements will be able to solve the discrepancies between our results and the results in [Helbing *et al.*, 2005]. These improvements are presented in this section. The section is divided into three parts: A discussion of the effects of modifying the interaction forces, a discussion of parameter values, and a discussion of possible errors related to our implementation of the simulation.

7.1 Modifying interaction forces

An obvious possibility when discussing why we do not see the expected results, is analysing the forces that comprise the model to see if we can identify any reasons for the discrepancy between the results we see and those we expected. As a starting point for this analysis, we look at additional forces presented in other articles, but that are not included in the model we have simulated.

We speculate that the reason that we do not see the faster-is-slower effect is that our model does not take into account frictional effect between pedestrians. Such a force has been suggested as a cause of the faster-is-slower effect in [Helbing and Johansson, 2009] and a term for the frictional force is given in [Helbing *et al.*, 2000], where they reproduce the faster-is-slower effect through simulations.

The friction force is given as a force which is perpendicular to the direction of the repulsive force, as illustrated in figure 14. The magnitude of the frictional component depends on both the distance and the relative velocity between the two pedestrians and would make it harder for pedestrians to move in dense crowds.

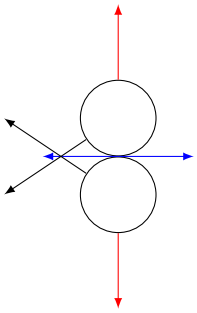
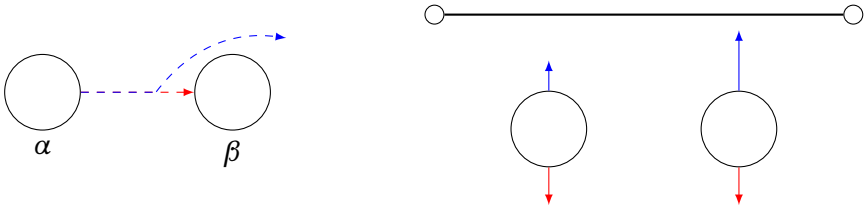


Figure 14: Friction forces (blue) are perpendicular to the repulsive forces (red) between two pedestrians and make it harder for pedestrians to move in crowded areas.

When looking at the simulations, there are two types of behaviour that we find presents us with problems: pedestrians walking directly towards each other do not try to avoid colliding, and pedestrians running towards a wall decelerate at the same rate as pedestrians walking towards it. These types of behaviour are illustrated in figure 15.

Inspired by the concept of frictional forces, we propose a way of making pedestrians avoid colliding with each other. This could be achieved by adding a force that is perpendicular to the direction of other pedestrians, much like the friction forces, but weaker and with a longer range. This might allow pedestrians to sidestep each other, as illustrated in figure 15a. We believe this would improve the results from the simulations of corridors that have pedestrians walking in both directions. It could also enable pedestrians to navigate around static obstacles, such as pillars. While we have not done any simulations involving such obstacles, results that show forces perpendicular to the direction of obstacles help pedestrians navigate around them have been demonstrated elsewhere [Pelechano and



(a) Pedestrian α walking towards pedestrian β.

(b) Pedestrians moving towards a wall at different speeds.

Figure 15: Problematic behaviour. (a) A pedestrian, α walking towards another pedestrian, β, does not divert to the side (red line), even though doing so would avoid a collision (blue line). (b) Pedestrians heading towards the wall at different speeds (blue arrows) are repulsed equally (red arrows).

Badler, 2006]. Finally, a perpendicular force might allow pedestrians to navigate along a wall if its target is on the other side of it; this could be used to simulate pedestrians trapped in a smoke-filled room where the exits are not visible.

The other problematic behaviour in our simulation is that when pedestrians approach a wall, they decelerate at the same rate no matter their velocities. In some extreme cases (i.e. when they are moving very fast) they will not decelerate quickly enough, and so will move through walls or even each other. We have tested whether lowering the time step size will alleviate this behaviour, and have found that it does not. We have seen in [Johansson *et al.*, 2007] that adding a velocity-dependent component to the wall repulsion force so that pedestrians moving faster will be repulsed stronger or at a greater range, could be a remedy for this behaviour. Adding such a force has been seen to improve model predictions when compared to data from real life situations [Johansson *et al.*, 2007].

7.2 Parameter values

The behaviour of the model depends heavily on the values of the model parameters. This means that choosing proper model parameters is important to get good results. As outlined in section 4.2, most of the parameter values we use, we have obtained from [Helbing *et al.*, 2005], however not all parameters are defined in this article, so we have had to find them elsewhere. Also, we cannot be sure that the implementation of their simulation matches ours closely enough that we can use the same parameter values. We have attempted to adjust some of the parameters when we have seen strange behaviour, such as pedestrians walking through walls. The fact remains, though, that we cannot be sure that the parameters we have ended up using are good ones, and this might be a source of error.

In newer articles we have seen parameter values obtained by fitting model predictions to results obtained experimentally, by filming real human crowds. We believe this is a good way to estimate parameters, but it leaves us at a disadvantage since we are not able to do such experiments with our own implementation of the model and we cannot compare the implementations to that of others to transfer the results. Therefore, selection of parameters continues to be a possible source of errors for our simulations.

7.3 Our implementation of the simulations

We have implemented the simulation of the model as closely to the description of the model as possible. However, as we have seen in section 4, there has been some areas where we have had to fill in details

that have not been explained in the articles describing the model. Indeed, in the formulation of the model itself, we have been forced to add features from different articles because the description in the original article was insufficient. It is possible that some of these necessary additions have been done differently than in the simulations performed by the authors of the original articles, and that this contributes to the discrepancies between their results and ours.

Of course we cannot completely rule out errors in our implementation either; while we do not believe any obvious errors exist, we do not have anything to compare it with that has the same level of detail. The only thing we have to compare our implementation with, is the code underlying [Helbing *et al.*, 2000], which the authors have published on their website. However, this code is quite inscrutable, so it would require considerable effort to analyse it and compare it with our own. A cursory glance indicates that there are features in it that we do not have in our implementation; whether these are vital or not we cannot say.

Finally, the use of random values for initial conditions (in the setting of pedestrian starting positions, size and initial desired speed) is a possible source of error. While we set a mean value and a standard deviation on the generated values, in some cases extremely high or low values are generated. An example of this is seen in figure 16, where one pedestrian starts out with a very low desired velocity, causing it to move extremely slowly. The long leaving time in this case is not caused by clogging of the exit, but simply by the delay from the time it takes the pedestrian to cross the distance between its starting point and the exit. To mitigate the effect caused by extremely high or low random numbers, we have sought to pick seed values for the random number generator that do not result in such numbers.

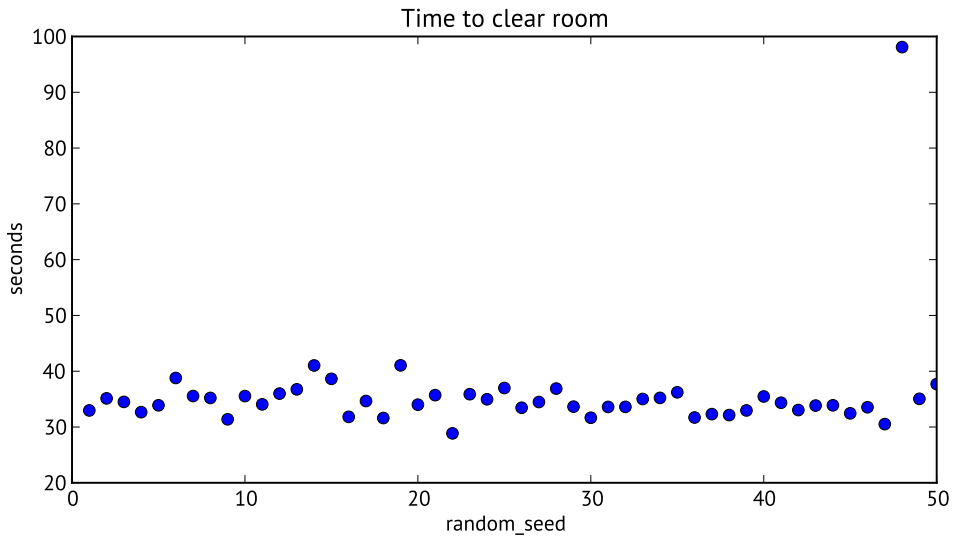


Figure 16: Leaving time for different random seeds. Results are dependent on random numbers; for a seed value of 48 (top-right corner), one pedestrian starts out walking very slowly impacting the total leaving time.

7.4 Summary

We have discussed several possible causes for discrepancies in our model, and possible remedies. The forces in the model only have a repulsive component that depends on the relative positions but not the relative velocity between pedestrians. This could be remedied by adding perpendicular forces and forces that are dependent on walking speed, which might produce better results. Depending on the way they are formulated, such forces could be interpreted as frictional forces between agents. Selection of parameters is also a possible source of error; a possible remedy, that is inaccessible to us, is obtaining

parameters from real-world experiments, by fitting data to simulations. Finally, errors arising from the nature of our implementation might occur. This includes possible coding errors, and errors as a result of extremely small or large random numbers used when setting initial conditions. This can be somewhat mitigated by testing the implementation, and by discarding very small or large random numbers.

In the next section we will assess social force models in a more general sense, and give our opinion of the state of social force models.

8 Assessment of social force models

In this section we will present our assessment of social force models in general, based on our work with the specific model and the results we have obtained. Based on our results, we will outline the strengths and weaknesses of social force models in general, and conclude by assessing how well these types of models simulate crowd behaviour.

8.1 Strengths of social force models

The main advantage of using social force models for modelling crowds is that it allows us to simulate a complex system using a quite simple formulation of the model. Simulating a large crowd of pedestrians using a classical (i.e. non-agent based) model would be very complex, and quite possibly impractical.

This simplification is possible because the complexities of the model behaviour is moved from the formulation of the model and into the calculations. That is, a very large number of calculations are needed to get any meaningful results out of this model. This means that working with this kind of model would be impractical without the help of computers, and indeed a large quantity of processing power is necessary to get any results. In our simulations this has been most apparent in the need to implement the calculation-intensive parts of the model in the C programming language, to be able to achieve reasonable computation times.

Another advantage of the simulations we get from the model, is that it is very straight forward to inspect the results visually, because it is possible to create drawings of the simulation steps. This means that comparing the simulations to e.g. videos of real-life crowds, and spotting effects such as the lane formation becomes trivial. It is also an advantage that simulations can be run in real time, so making visual assessments of results is easy.

Of course having a simple model that is practical to implement is of no use if it does not give useful results. Empirical studies have shown that social force models are able to show phenomena that correspond to real crowds, and as such do provide meaningful results in some cases [Helbing *et al.*, 2005; Helbing and Johansson, 2009].

8.2 Weaknesses of social force models

While social force models in some cases have been shown to give results that correspond to real life observations, there are several weaknesses to this approach to modelling crowds. Some of these weaknesses are related to the way the models are presented, and some are more fundamental to the nature of social force models.

When working with social force models, we have had to piece together a working model from several different sources. This exposes a difficulty in assessing the models: It is not always obvious if a given weakness is due to an inherent quality of the models, or if it is simply due to a weak or missing formulation of some part of it. Especially precise results of simulations have been difficult to find, and as has been shown in section 4, we have had to fill in several details ourselves. As mentioned above, the difficulty in estimating parameters has also provided a barrier in this respect. In addition, as mentioned in section 4.2.5, it is difficult to assess whether a simulation run provides meaningful results, so determining which parameter values work further adds to the difficulty of evaluating the model.

Setting aside the difficulties in finding detailed information about the model, it is readily apparent that social force models are in a relatively early state of development, so the amount of empirical data available to assess the quality of the models' predictions is quite low. This means that even if the social

force models have shown some promising results, it is impossible to say with confidence that they do indeed predict the actual behaviour of crowds very well.

Using the distinction between simulation-based models and theoretical models laid out in [Jensen, 1980], it is clear that social force models belong in the category of simulation-based models. That is, even though the notation and concepts are borrowed from physics, there is no underlying theory behind the formulation of social force models. Instead, it is a model that is formulated to simulate a concrete set of observations of crowd behaviour. This means that in order to build confidence in the model's ability to provide useful results, it has to be shown to conform to empirical data with a high confidence. Something that these models have not achieved, at least so far.

Further compounding these uncertainties is the fact that the model is formulated in a way that is counter-intuitive to the way human behaviour is normally perceived. One of the reasons we are sceptic that these forces are able to completely explain human behaviour, is that they pertain to behaviour of objects in a physical world, but they do not obey the traditional physical laws of motion. Additionally, that human behaviour is reducible to simple repulsive forces is counter to what we believe is reasonable. This means that if this is indeed the case, strong evidence is needed to convince us, which is not currently provided.

Finally, further cementing the social force models' status as simulation type models, is the fact that different variants of social force models use completely different parameters and formulations of the different forces, sometimes even contradicting each other, making it apparent that the models are changed in arbitrary ways to better match empirical observations. This belief is corroborated by the fact that the models do not make any new predictions that are then tested, but instead only seem to attempt to replicate already observed behaviour.

8.3 Conclusion on the assessment

Weighing the advantages and disadvantages of social force models against each other, it is quite apparent that the models do not instil a strong sense of confidence in their predictions. However, the crucial advantage that these models have, is that they in some cases are able to provide reasonable simulations of crowd behaviour, that no other models (that we have seen) have been able to. This means that while social force models are far from perfect, they are in many ways the best available way of evaluating e.g. a new building's suitability for efficient crowd movement. And while the models may not be able to provide any underlying theory or reason for the crowd's behaviour, in practice they may, given further adjustments and experiments with real life observations, be a substantial improvement over today's standards.

9 Summary and conclusion

We have created a computer simulation of a social force model in order to study social force models in general and find out how well social force models simulate the behaviour of crowds, and what strengths and weaknesses of these models are.

In order to get an overview of the different social force models, we have first examined the field of social force modelling and presented the variations that exist, as well as some of the results that are presented in the literature. We have picked an exemplary model that is described in detail in the article that presents it, and analysed it in detail. In this process we have had to fill in details from articles other than the one that presents our chosen model.

Based on the analysis of the model, we have implemented our simulation of the model. While doing this, we have had to fill in some details in order to go from the abstract model formulation to a concrete numerical simulation. These details include how to approximate the movement of pedestrians, how to set initial conditions and values, and how to implement the interaction between pedestrians and walls in practice.

From our results, it is clear that we have successfully implemented a running model simulation, that (with the right parameters) exhibits reasonable pedestrian behaviour upon visual inspection. We have not done an exhaustive review of the different parameters' effect on the simulations, however, since we have deemed doing so manually impractical, and we have not been able to come up with a way to automate it. As such, we are not able to say anything conclusive about the parameters' effect on model results.

We have attempted to replicate the results we have seen in the literature. While some results have been successfully replicated, other effects do not manifest themselves in our results. We have discussed several reasons for this discrepancy, including features that are lagging from the model, parameter values, effects of using random numbers to generate the initial conditions and possible errors in our implementation of the model.

Based on the results of our own simulations and our review of the social force modelling field, we have assessed social force models and their strengths and weaknesses. We conclude that social force models are not built on any theories for the behaviour of crowds, but are created to replicate a set of observations. As such, any confidence in their predictions must come from a record of producing results fitting observations; and since the field is relatively new, they have not yet reached this state. Social force models do, however, provide a practical way to simulate something that would otherwise be impossible to simulate. As such, they are the best available way to provide e.g. guidance when designing facilities that must accommodate many pedestrians, and given time the accuracy of their predictions will probably increase.

9.1 Further work

We see three main directions our work could be continued in: adding new features to the model, examining parameter interdependence and the effects of different parameter values on the model behaviour and testing our model implementation on real world observations.

In section 7.1, we discuss features that we believe would improve the results obtained from the simulations if added to the model. A possible avenue for further work would be adding some of these features and examining what impact, if any, they have on the model behaviour. Among the evaluated features could be substituting another way to approximate pedestrian movement than using Euler's method as we do.

Another possible expansion of our work is coming up with a way to automatically test whether a simulation run is viable (i.e. doesn't break down due to pedestrians walking through walls etc.), and using this to test different parameter settings to discover which parameters are interdependent, and how different parameter setting affect the viability of the simulations.

Finally, we think it would be interesting to compare the results of our simulations with real world observations. While some articles compare the results of simulations with real world observations, these comparisons are specific to the simulation implementations used in the article (the details of which are not always available). We think it would be interesting to test our implementation of the simulation against real world observations.

10 References

Arciniega, Armando and Allen, Edward, 2003. "Rounding error in numerical solution of stochastic differential equations". *Stochastic Analysis and Applications*, 21(2):281–300.

Butcher, J., 2003. *Numerical Methods for Ordinary Differential Equations*. J. Wiley, London. ISBN 0471967580.

Ceperley, D. M., 1999. "Microscopic simulations in physics". *Reviews of modern physics*, 71:s438–s443.

Frenkel, Daan and Smit, Berend, 2002. *Understanding Molecular Simulation*. Academic Press.

Haim Levy, Moshe Levy Sorim Solomon, 2000. *Microscopic Simulation of Financial Markets*. Academic Press.

Helbing, Dirk, Buzna, Lubos, Johansson, Anders and Werner, Torsten, 2005. "Self-organized pedestrian crowd dynamics: Experiments, simulations and design solutions". *Transportation Science*, 39:1–24.

Helbing, Dirk, Farkas, Illés, Molnár, Petér and Vicsek, Thamás, 2002. *Simulation of Pedestrian Crowds in Normal and Evacuation Situation*. Springer.

Helbing, Dirk, Farkas, Illés and Vicsek, Tamás, 2000. "Simulating dynamical features of escape panic". *Nature*, 407:487–490.

Helbing, Dirk and Johansson, Anders, 2009. "Pedestrian, crowd and evacuation dynamics". *ETH Zürich*.

Helbing, Dirk and Molnár, Péter, 1995. "Social force model for pedestrian dynamics". *Physical Review*.

Helbing, Dirk, Molnár, Petér, Il and Bolay, Kai, 2001. "Self-organizing pedestrian movement". *Environment abd Planning B: Planning and Design*, 28:361–383.

Huth, Andreas and Wissel, Christian, 1992. "The simulation of the movement of fish schools". *Journal of Theoretical Biology*, 156:365–285.

Jensen, Jens Højgaard, 1980. "Matematiske modeller er væsensforskellige, selvom de ser ens ud". *Tekster fra IMFUFA*, 26.

Johansson, Andreas, Helbing, Dirk and Shukla, Pradyumn K., 2007. "Specification of a microscopic pedestrian model by evolutionary adjustment to video tracking data". *Advances in Complex Systems*, 10:271–288.

NumPy, 2010. "Scientific computing tools for Python - NumPy".
URL `http://numpy.scipy.org`

Pelechano, N., Allbeck, J.M. and Badler, N.I., 2007. "Controlling individual agents in high-density crowd simulation". *ACM SIGGRAPH/Eurographics SCA*.

Pelechano, Nuria and Badler, Norman I., 2006. "Improving the realism of agent movement for high density crowd simulation". *ACM SIGGRAPH/Eurographics SCA*.

PyGame, 2010. "PyGame - python games development".
URL `http://www.pygame.org`