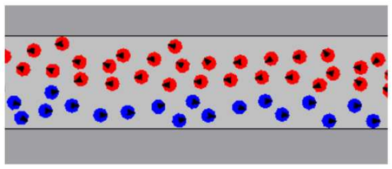
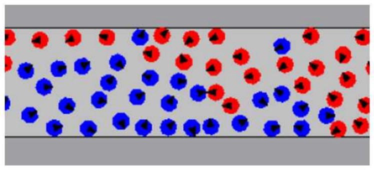
A Stream algorithm for crowd simulation to improve crowd coordination at all densities.

Arthur van Goethem

29th August 2012





Department of computer science

Utrecht University, The Netherlands

Supervisor: R. Geraerts

ICA-0215872

Abstract

Crowd simulation is becoming an essential part of computer sciences. In the last decades the research field has taken a flight. Current state-of-the-art micro simulation algorithms for crowd simulation are able to deliver very convincing results. Recently, vision-based algorithms have been developed that are based on the field of vision of individual agents. These algorithms allow agents to interact realistically at low to medium densities, but have trouble coordinating movement at high density. A concurrent development are flow algorithms. These algorithms are able to solve high density scenarios due to an inherent high level of coordination, but lack individuality of agents.

Up till now, however, no algorithm exists that can handle both low densities as well as extremely high densities realistically. In this thesis we will define an algorithm that can handle the entire density spectrum. Agents are individual entities with personal goals and settings. The algorithm will incorporate the implicit, local coordination present in real-life crowds.

We will show that coordination is essential in high density areas. Information is implicitly stored in a crowd's movement. We introduce the concept of streams to extract this information. The implicit global coordination of streams improves crowd flow and supports lane formation in extreme densities. We will implement this theory to achieve an algorithm that can handle all different densities.

To allow for both individual as well as coordinated behaviour we will introduce the concept of incentive. An agent's incentive deterimines if it will display more individualistic behaviour or be group-oriented. This allows agents to display individualistic behaviour as long as possible, while still cooperating if necessary.

Experiments show that the streams algorithm together with our incentive-based interpolation scheme are a valuable addition to current micro collision-avoidance algorithms. Our algorithm is able to handle higher densities correctly, whilst maintaining behaviour at low densities. The inherent coordination at high densities is missing in all current algorithms. Even at lower densities the streams algorithm contributes to quicker and clearer lane formation. Our algorithm is able to function properly at all densities.

This project has been made possible by a collaboration between Utrecht University and INCONTROL Simulation Software. All algorithms have been implemented in Pedestrian Dynamics, which is a pedestrian flow simulator by INCONTROL.

Contents

1 Introduction

In Section 1.1 and 1.2 we will give a short summary of what crowd simulation entails. We will give a description of our project goals in Section 1.3. Readers familiar with the concept of crowd simulation can start with this section. The overall structure of this thesis is explained in Section 1.4.

1.1 Introduction

Crowd simulation is a broad topic. Trying to define it clearly is a hard problem. We will give an all-encompassing definition. This implicitly implies that we can not define all parts exactly. Our goal will be to give a global overview.

The most basic definition of crowd simulation entails the simulation of a crowd. However, crowds show very diverse behaviour in diverse situations. And, even in comparable situations, different behaviours may emerge. Crowds, in turn, consist of individual (human) beings, who also display diverse behaviour. An already broad set of circumstances is extended even further as different approaches to crowd simulation are necessary. Different types of problems require different types of answers.

Generally, crowd simulation entails realistically mimicking, in a virtual way, the behaviour of a crowd of entities in a diverse set of situations. Depending on the problem, we might want to know the average density of a crowd at different places of a museum tour. Or we might want to know the exact position of agent 'John', who is currently residing in a giant corn maze. Crowd simulation is applicable in many different situations and is these days a part of everyday life.

Definition 1 (Crowd simulation) *Crowd simulation entails realistically mimicking, in a virtual way, the behaviour of a crowd of entities in a diverse set of situations. The behaviour of the entities should be consistent with observed behaviour in real-life.*

Computer models are a clear-cut example of crowd simulation. Crowd simulations are run to test evacuation procedures, increasing safety. The design of a building is tested beforehand to prevent unexpected throughput problems. But we might also think of training security personnel for an upcoming large scale event. An entirely different area of use are computer games. Think of the birds taking to the air as soon as a shot is fired in a 3D shooter game or the squads being controlled in the latest Real-Time Strategy. All of these are likely controlled by some form of crowd simulation. We have not even mentioned the movies yet. In short, crowd simulation is everywhere these days.

In Section 1.2, a brief description of the main components in all crowd simulations will be given. In Chapter 2, an in-depth discussion of the subjects directly related to the problem discussed in this thesis is provided.

1.2 Different components of crowd simulation

Crowd simulation is a large topic that covers many areas. As a result, many different sub-problems are part of crowd simulation. Each of these sub-problems is an active research field on its own. As a consequence, crowd simulation is a field that is constantly changing and growing. In this section we will give a basic description of the main problems crowd simulation spawns. A more in-depth discussion of the subjects directly related to this thesis will be given in Chapter 2.

1.2.1 The path-finding problem

One of the main problems when simulating a crowd is how to route all entities. Routing is already a hard problem for a single entity. The problem of path-finding consist of finding a valid

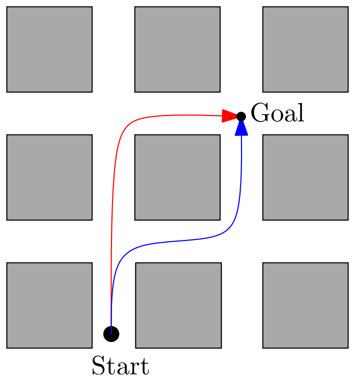


Figure 1: A grid-based city

route for a given entity. The route should lead the entity from its current position to its desired location, anywhere in the environment. Not all routes are allowed however. There are many homotopically different routes to take and the algorithm should compute an (optimal) choice among these. An example of this is a grid-based city (Figure 1). We could imagine a route consisting of a right turn followed by a left turn. But a route consisting of taking the second turn right is also valid. Both routes will get the agent to the same intersection. General routing is less clear than a grid-based environment and requires a more abstract way of thinking about the environment.

The problem becomes even harder when we are trying to route a multitude of agents. Agents should avoid the static obstacles in the environment and find their way around them. At the same time, they should also be able to avoid collisions with each other. Avoiding these collisions requires agents to stay away from each other.

Yet in the movement of other agents, knowledge about the environment is stored. Different agents have knowledge of a different subset of the entire environment. This knowledge is expressed in their choice of movement. An agent might want to make use of this knowledge to complement its own knowledge. This type of knowledge transfer is quite common. A recognizable example entails following the feet of the person ahead in a dense crowd. The underlying assumption is that they will know more of the environment ahead. This type of knowledge requires agents to make a coordinated movement, in contrast to avoiding other agents. There is an implicit trade-off between this coordinated movement and 'egocentric' collision avoidance. This trade-off will be the main topic of this thesis.

1.2.2 Simulation scales of crowd simulation

Once agents have decided on a global route, they know how to get to their goal. Whilst moving towards their goals they have to avoid each other as well. How this is handled locally depends on the scale of the simulation. The scale determines in a large part what kind of information can be retrieved by the crowd simulation. Three different scales are considered in both crowd and traffic simulation, namely *micro*, *meso* and *macro* scale [6, 7, 8].

Crowd simulation on a micro scale is the most detailed level. Each agent pursues its own goals and moves in the most realistic way possible. It will try to avoid collisions with other agents while doing so. On this scale an agent is determined by an exact position and velocity. With this knowledge agents can prevent collisions and determine if they need to slow down if dense situations require them to. Simulations are consequently nearly collision-free. An example where this kind of information is relevant are computer games. Every agent in this setting should avoid each other. It is not acceptable to have the cavalry walk straight through the infantry.

On the meso scale we are not necessarily interested in the exact position of an agent. Agents

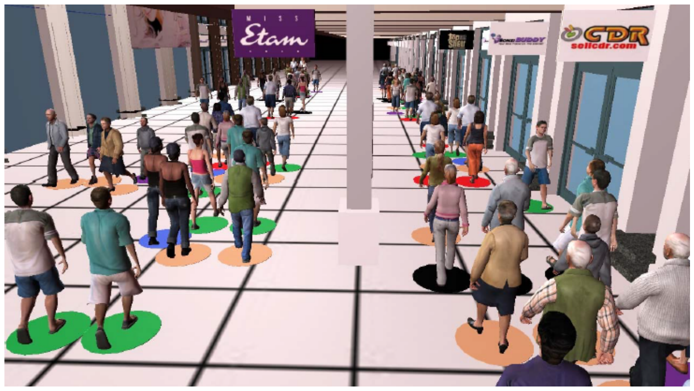


Figure 2: Movement of groups within a crowd. Group movement increases realism in crowds. Deformation of groups is an active field of research at the moment.
(Figure courtesy of 'Simulating and evaluating the local behavior of small pedestrian groups' [11] by Karamouzas)

still have their own goals and are still represented as particles. However these particles only describe an approximate position of each agent. We are sure that the agent is likely to be somewhere in the vicinity of this position. Collision avoidance is less relevant. Agents simulate the effect of this avoidance behaviour by altering their speed depending on the local density. As collisions are unimportant, all agents could simply move along the same routes. The meso scale is related to the kinetic theory of gasses, where particles are still part of the model, but interaction is represented by a statistical process [9]. This scale of crowd simulation can be used for measuring throughput of agents and density buildup. An example could be testing where high densities occur in a guided museum tour.

At the macro scale we even drop the concept of agents as particles and instead represent the crowd as a fluid with specific densities at different positions in the 'crowd'. Individual agents no longer exist. A distribution is set over the different goals this crowd might have. Density is only measured along complete edges[1]. These known densities together with the liquid properties of a crowd describe a flow of the crowd following the theory of fluid dynamics. This kind of macro simulation may be found in simulations where we want to simulate very large numbers of agents who are consistently forming dense flows. We are only interested in the predicted distribution of densities and whether local buildups of extreme density occur. An example might be simulating the Hajj pilgrimage [10].

We are mainly interested in the micro scale in this thesis. We want to have realistic behaviour at a local scale as well as on a global scale. Agents should avoid all collisions. We will show that inter-agent coordination is an important factor in this at high densities.

1.2.3 Social interaction and behaviour

Visualisation is an important part of crowd simulation. We will not discuss 3D-graphics in this thesis, as this is not necessarily part of our focus. However, visualising social interaction is important for believable crowd simulations. As humans are social creatures, crowd simulation is a social process. This thesis does not take into account this kind of social behaviour, except as a possible extension. Therefore, we will mainly refer the interested reader to related research.

Most crowd simulations make the assumption that all agents walk alone. In real life, however, this is most certainly not the case. Humans form groups, and, more often than not, move in groups as well. How these groups react to other humans and deform is one of the main

[1]An edge in this context is defined as a route starting at one intersection and ending at the next intersection.

topics of research at the moment. Research into this kind of social behaviour has e.g. been done by Karamouzas and Overmars [11] based on the paper by Moussaïd et al.[12] (Figure 2). They present some base formations for groups. Groups alternate and interpolate between these formations, depending on the environment.

Qiu and Hu [13] define an inter- and intra-group matrix describing how agents within and in-between groups influence each others movement. Takahashi et al. [14] use a weighted graph to represent group formations. These graphs are interpolated in order to transform from one formation into another. A Cellular Automata based approach is given in the paper by Sarmady et al. [15] .

Another important social phenomenon is the social behaviour and formation of groups. Displayed behaviour by agents affects the behaviour of other agents. This can lead to unexpected group behaviour. We might think of riots as a basic example. Social effects can cause an otherwise docile crowd to turn into a riotous, vandalizing mob. The thesis by Wijermans [16] discusses this psychological approach to crowd simulation more in-depth. In his paper, Granovetter [17] describes the effect of a threshold model on crowd behaviour.

Often crowd simulations, no matter how realistic, look unlively. This is partly because agents show no social behaviour at all or unfitting behaviour. It is also partly because all agents appear to be behavioural clones. Both issues are currently being addressed. In the paper by Guy et al. [18] a proposal is made for ascribing different personality traits to different agents. Their extension creates more diversely populated crowds. Agents can have different movement strategies. A hurried agent might pass other agents sooner than a shy agent.

In the paper by Lerner et al. [19] different behaviours are fitted to pedestrians depending on the correlation of the situation with real-life situations. Thus agents seem to display more natural context-related behaviour. For example, conversing with their fellow group members. Both approaches seem to generate more life-like crowds.

In this thesis we will not consider any of these aspects of crowd simulation. Our model is able to facilitate these extensions. In future work these extension might be included.

1.3 Project goals

In this project we will mainly be concerned with micro simulation behaviour. Our goals for this project will be twofold.

In many crowd simulation algorithms either individual behaviour or crowd flow propels an agent. Individual behaviour allows each agent to traverse its own path. Every agent in the simulation pursues its goal independently of what other agents are doing [2, 20, 21]. However, this individual behaviour limits crowd movement at high densities. Inter-agent coordination is essential at these densities.

Crowd flow is the exact opposite of this. Agents pursue group goals and local motion is coordinated between agents. Individual behaviour is limited as only entire groups can show different behaviour. A good example of this are flow fields [22, 23, 24, 25]. A local neighbourhood is often defined through the use of a grid-based structure. Within this local neighbourhood a common movement is decided upon.

Nearly all crowd simulation algorithms are based on either method. Agents are thus forced to commit to one strategy. In real-life situations we would like agents to be able to seamlessly switch between the different strategies. In this thesis we will attempt to merge both approaches and automatically apply the correct combination of both. This will lead to more realistic behaviour. This switching behaviour will mainly be dependent on the local density. Local density is defined as the percentage of the local area covered by agents. We will list several options to implement this local behaviour and discuss advantages and disadvantages.

Agents will be able to follow their own route. If density increases they will start to move along with local *stream*. How to go along with the stream and in what way to switch between

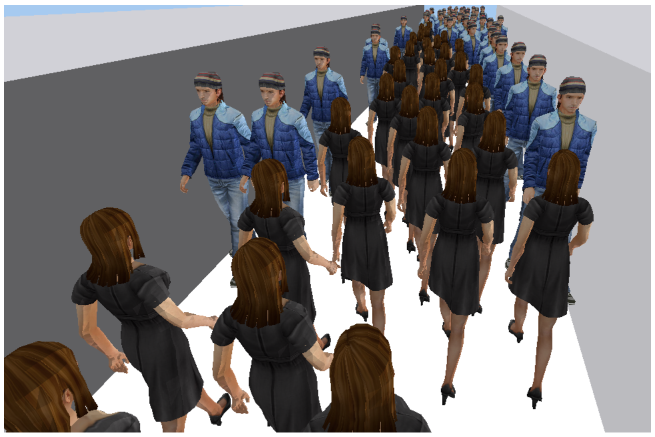


Figure 3: A high level of group coordination is required in dense areas. This coordination leads to emergent behaviour such as lane formation in very dense scenarios and decreases the total energy spent. Information about the local situation is implicitly transferred between agents. *(The figure is a screenshot taken in Pedestrian Dynamics [26] running our stream algorithm)*

both approaches will be our main contribution.

We will propose a sliding approach in which agents are influenced more by the crowd as density increases. This sliding approach will allow for a smooth transition between both strategies. Agents will gradually be more affected by the crowd and no clear transitions will take place.

The second goal this thesis will pursue is to allow individual behaviour with respect to this strategy choice. Each agent should have its own meta-strategy. It should be able to make an individual choice between both strategies. Some agents might prefer coordinating with crowd motion at all times. Others, in the same situation, might decide to move upstream of a crowd. Different profiles can be identified for these meta-strategies. An clear example of the second kind of profile would be a police officer. Police officers will be more determined to reach a certain destination than most agents might be.

For the global path planning of agents we will base our work on the multi-layered Explicit Corridor Map (ECM) by van Toll et al. [27]. The multi-layered ECM is an extension of the ECM [28]. The ECM gives an exact description of the walkable space around a chosen path. This allows for the full range of evasive manoeuvres which might be taken by the agents. The multi-layered ECM gives the same description in a 2.5D environment. Such an environment consists of several 2D layers connected by a set of shared line segments [27].

We will use the Indicative Route Method [5] (IRM) for agents to determine basic routes through these corridors. The IRM creates smooth and realistically looking routes based on an indicative route through the environment. Eventual paths for agents will be based on these desired routes in combination with evasive and flow behaviour. A more extensive explanation of the ECM and IRM is given in Chapter 3.

Density-based path planning [29] will ensure agents pick efficient global routes. During global path planning, local density is taken into account. The higher the local density, the higher the cost to travel along this route. This simulates the effect of longer travel times due to higher densities. An agent will pick the route with the lowest expected travel time based on the current situation. While moving along the planned path agents replan their routes taking into account all currently visible density information.

This project has been supported by a collaboration between Utrecht University and INCONTROL Simulation Solutions [30]. The algorithms which have been developed, have been integrated in Pedestrian Dynamics. Pedestrian Dynamics is a new crowd simulation program developed by INCONTROL, which allows for quick yet extensive modelling of 2.5D environments and subsequently simulating and evaluating crowd movements through these environments.

1.4 Document structure

The rest of this document is organized as follows. In Section 2 related work on crowd simulation is surveyed. We will start at the top-level of global path planning and zoom in till we reach the micro simulation level. We will end in Section 2.4 by describing how this thesis will extend current knowledge on micro-level crowd simulation. In Chapter 3 we will describe the preliminaries to our work.

The following chapters will describe the different aspects of our core research. In Chapter 4 we will discuss our new approach for moving agents in a dense crowd. In Chapter 5 we will discuss how individual and group behaviour can be merged to obtain an complete algorithm for low up to extreme densities. The algorithm will maintain individuality between agents and allow for individual goals.

The experiments run are described in Chapter 6. In Chapter 7 we will discuss the results obtained. Chapter 8 will state some possibilities for future work. An overview of the entire Streams model is given in Chapter B. Finally in Chapter C will discuss important implementation details of the different density approaches tested.

In the addenda we will discuss how the main vision-based algorithm by Moussaïd et al. [20] was optimized for use in our thesis (Chapter D). Chapter E in the addenda will discuss some of the best practices we came across while implementing our theory in practice.

2 Related work

In the context of crowd simulation much research has been done. We will give an overview of the subject in this chapter. We start off at a global perspective and work our way down. In the last subsection we point out how our research extends current knowledge about micro-level crowd simulation.

2.1 Global path planning

When routing an agent through an environment we are charged with two different tasks. The first one is finding a global route from the start position of the agent to the desired goal position. If we had a map of the environment, this would be relatively easy.

The most simple way to create this map is to overlay a grid on the environment (see Figure 4). We treat each cell as connected to its four neighbours. Vertices related to cells containing obstacles are removed from the graph. The resulting graph is a representation of the free cells and consequently of the free space. We can now apply a basic search algorithm on the graph. The book of Lavalle gives an excellent overview on this topic [31].

There are, however, a lot of disadvantages to this way of representing an environment. This is mainly due to the way the graph is created. No knowledge of the environment has been implemented in the process. Consequently, there are many superfluous arcs, making storage expensive. Other problems occur due to resolution problems. A cell covering a narrow path can be marked as occupied as it partly overlaps obstacles. This causes narrow paths in the environment to be missed in the resulting graph. It might make planning a path impossible. Even if a path is found, it is still of a low quality. Paths can only take sharp, perpendicular turns due to the grid-based structure. This makes them unrealistic for human motion. Routing multiple characters is even harder, as multiple paths are hard to coordinate. Any realistic path planning graph will have to be created by incorporating knowledge of the environment. The graph should be as sparse as possible and should succinctly describe the environment.

The concept of a roadmap addresses this problem. A roadmap is a graph that connects to each point in the environment through a simple path[2]. Besides this, the graph should also consist of solely one connected component.[3] If the graph adheres to these properties, we can connect any point A in the environment with any other point B through this graph. From point

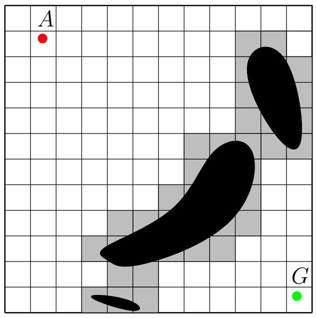


Figure 4: An example of grid-based path planning. Even though the agent A can reach the goal G no path will be found. All grey cells are supposed to be occupied by obstacles and, hence, are not traversable.

[2]A simple path should be as simple as not to require any explicit routing.

[3]It is possible for a roadmap to consist of multiple components. However, these components should always be connected at certain points on the graph. This can either be through a direct connection or through a simple path. Without these connections it is not guaranteed that we can compute a route between every pair of points. We can represent these relations by a (one-way) arc, which will leave us with a single connected component.

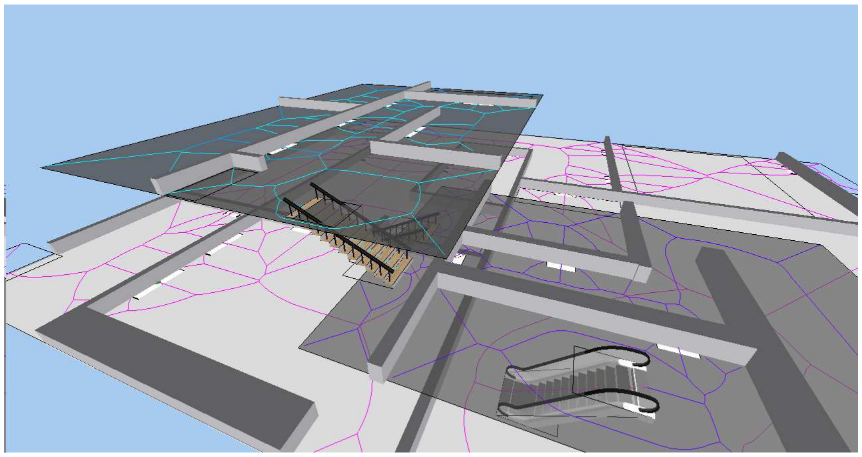


Figure 5: An example of a multi-layered Explicit Corridor Map (ECM). The Medial Axis of the ECM is represented in green, purple and pink for the different layers.

A we can find a simple path to the graph. By travelling over this graph we can get to a point C on the graph that will have a simple path to the goal position B. Thus we obtain a global path for the agent.

Many different techniques exist to create this graph. The book of Lavalle [31] gives an excellent overview on this topic. Usually the A*-algorithm [32] or Dijkstra's algorithm [33] is used to compute the required path through this graph. We will not dive deeper into this subject as it will not affect our main topic.

There is one problem with the described techniques up to now. If an entire crowd follows the same graph, agents will end up opposing each other on some edge at some time. These agents should be able to pass each other without colliding with each other or with the environment. The main problem here is that we are still unclear as to the exact shape of the environment, even though we know the global connectivity. Thus for agents trying to pass each other on an edge, it is unclear how much space they have to avoid each other and where this space might be.

Navigation meshes describe the entire walkable space. Instead of just describing possible corridors by a single arc, more information is stored. Even though memory requirements are slightly higher, this extra information allows algorithms to compute the entire free space. A simple approach to store the representation of this free space is through the use of cylinders centred on the medial axis [34, 35]. Each cylinder describe a section of walkable space without obstacles.

In the (Explicit) Corridor Map (ECM) method by Geraerts [28, 36] a new data structure is proposed that more efficiently stores a representation of the entire free space. For each path a corridor is extracted that entices the entire free space along this path. This corridor is described by annotating the closest obstacle points when the global shape of the environment changes. The global structure of the environment is represented by the medial axis (see Figure 5). A more complete explanation is provided in Section 3.1. This representation allows agents to avoid each other when traversing along the same edge and still remain in a valid position.

Having described the entire free space, it is no longer clear where the agent should plan its route. A likely candidate would be the medial axis of the corridors. Nevertheless, this can still cause strange, sharp angles in the computed path. The Indicative Route Method (IRM) by Karamouzas [5] solves this in order to create more realistically looking behaviour. An attraction point on the desired route is defined. This attraction point precedes the character and causes it to defer from its computed route to create more fluent paths. Boundary forces keep the character inside the corridor.

For our project we will use the ECM and IRM methods as global path planning methods.

2.2 Local path planning

Having decided on a global route, there still remains the second task required for routing a crowd, namely local path planning. Each agent following a global route still needs to take evasive actions regarding moving obstacles such as other agents. A number of different approaches to this problem are currently being investigated.

One of the earliest models designed to tackle this problem[4] is the boids model as described by Reynolds [37]. It is essentially an extension of the principle of a particle system [38] and describes the interaction between a group of birds. Each bird has different forces which act upon it. There are forces that cause it to remain as close to the group as it can, but also forces that push it away as it gets too close to another bird. A final force causes it to align its velocity to the neighbours. The combination of these forces causes the flock to show recognizable flocking behaviour.

This concept of non-physical forces is taken even further by Helbing who describes a social-force model for pedestrians [1]. In his model a set of social forces act upon each agent causing it to alter its velocity. These social forces can be seen as motivations to act for each agent and are not necessarily physical. An example of such a force would be preventing getting too close to some other, unfamiliar agent. The main drawback of social forces, as first described by Helbing, is that it causes agents to behave reactively. Only at the last possible moment will agents realign their course. This gives rise to unnatural behaviour.

In the paper by Karamouzas [2] the concept of social forces is used in combination with a predictive collision-avoidance (social) force, giving rise to smooth long term collision avoidance.

A different approach is the use of flows as described in the paper by Treuille et al. [22]. In their paper they give an algorithm based on continuum dynamics. For each group of agents with the same goal and same properties they define a force field. This force field indicates the required direction for all agents in each point of the environment, leading them towards their goal. It takes care of both global as well as local path planning.

The main advantage with this approach is that the force field only has to be calculated once per time step. Once computed, all related agents can instantly be updated. The disadvantage is that we need a force field for each of the different groups of agents. This can quickly lead to overhead when agents become less likely to have the same goals. As there is no coordination between different groups, agent movement is still far from optimal. Also individual properties of agents are lost.

A different approach are Cellular Automata (CA). This approach was given rise to by the field of artificial intelligence. In CA, the environment is discretised into a set of discrete positions through the use of a square-based grid. Agents occupy one of these cells and can transition to a new state where they occupy a nearby cell. As stated by Wolfram [39], CA are inherently a discrete idealisation of fluid flows. An advantages of the CA approach is that it makes collision avoidance relatively quick and cheap. It also gives reasonably realistic behaviour at low densities. At higher densities, however, the underlying grid is often a limiting factor. Agents are forced to move according to the grid, giving rise to unrealistic motions. This grid limits the number of directions available to agents as well as forcing agents into predefined positions.

Recent work in this direction has been conducted by Burstedde et al. [40]. They introduced the concept of a 'scent trail', causing agents to give rise to lane formations. Blue and Adler [41] describe three different cases of bi-directional flow and do research as to what the effect of

[4]More accurately, the boids model is a related problem. But as we shall see the connection is clear enough to authorize its mentioning here.

different parameters is on the behaviour of agents in these cases. Yue et al. [42] have tried to make agents more intelligent by introducing four basic probability rules. Combining these rules gives rise to a 'clever' probability distribution, causing agents to navigate through the crowd.

A continuous approach to the path-finding problem is proposed by van den Berg [3, 43]. The Reciprocal Velocity Obstacle (RVO) is an extension of the concept of Velocity Obstacles [44] from robotics. The RVO method enables agents to define a set of allowed velocities. All allowed velocities will lead an agent to avoid collisions with any other object (agent). RVO is a realistic and quick approach to collision avoidance and one of the more popular approaches. However, even RVO remains reactive. Agents only react if there is an imminent need to. Another disadvantage is that agents react to agents who are behind it, which may be deemed unrealistic in a majority of scenarios. RVO guarantees collision-free motion as long as this is possible. At high densities this may give rise to energy-inefficient, unrealistic motions, in contrast to other methods, such as, the predictive collision-avoidance method [2].

One of the newest areas of research is based on a more human-like approach to collision avoidance. As shown in the paper by Cutting [45], vision is the main source of information for agent collision avoidance. In vision-based algorithms, the environment within an agent's field of view is scanned for possible collisions. Based on these future collisions an appropriate reaction is selected. Pettré et al. [46, 21] try to mimic this natural behaviour as close to reality as possible. They have shown by comparative studies that vision-based algorithms give the most realistic behaviour. Moussaïd et al. [20] created a more efficient and heuristic way of determining the vision of an agent and its possible choices for altering its local route. Agents choose a least-effort path through their nearby environment. They weigh out the change of pace and deviation from the straight-line course towards the current goal. However, even though the algorithm by Moussaïd et al. is more efficient, computational complexity still remains an issue with vision-based approaches.

For this thesis we will require a local collision avoidance algorithm. In our experiments we will compare the approaches by Karamouzas et al. [2], Moussaïd et al. [20] and van den Berg et al. [3]. We will extend this algorithm to handle both low and high densities.

2.3 Dense crowds

As crowds get to a extremely dense state, emergent phenomena start to occur that cause danger to the crowd [47, 48, 49]. As crowds are never evenly packed, density measurements can give different results depending on the size of the measured area. Local density can reach peak values that are two times higher than the average density. On average, a density of six through nine persons per square meter is considered extreme and dangerous [48, 50]. In the safety guide for sports events published in the United Kingdom, a safety threshold is given of 4.7 persons per square meter [51].

In crowds of extreme densities, stop-and-go motions appear, which can be related to the stop-and-go motion in traffic jams. After a certain period of time these stop-and-go motions transfer into a state of crowd turbulence. In crowd turbulence, uncontrollable motions occur in all directions causing peak surges in local density and crowd pressure. Crowd turbulence is an extremely dangerous phenomena. In 2010 at the Love Parade in Duisberg, Germany, 21 people died and over 510 were injured. People entering a tunnel to get to the festival ground, were unable to see that the festival grounds were closed off. A fatal build-up of crowds occurred in the tunnels, leading to extreme densities and crowd pressure. In 1994, during the annual Hajj in Mecca, 266 people died during the 'Stoning of the Devil'-ritual at the Jamarat bridge. Extreme densities on the bridge led to crushing of people. There are many more examples where extreme crowd density has led to fatal consequences [52].

Even though quite some research has been spent on extremely dense crowds in real life [48, 53, 54], a remarkably sparse number has actually focused on extremely dense virtual crowds. In the article by Narain et al. [23], a representation of a dense crowd as a continuum fluid is given. On a staggered grid, the velocity of the crowd flow is given as if it were a real fluid. An agent uses this flow information to determine its next velocity. This is done by interpolating between the flow direction and its desired direction depending on the local density.

Curtis et al. [55] also study an extreme density scenario occurring during the annual Hajj. They use a finite state machine to determine an agent's state of mind. This information is added to the RVO method [3] as a determination parameter. This parameter determines the relative effort an agent invests to prevent collisions with other agents. More determined agents might force other agents to take more evasive actions, whilst taking less evasion measures themselves.

Yu and Johansson [56] extend Helbing's social-force model [1] to include a parameter describing physical agent interactions occurring at extreme densities. Their model seems to be able to represent the state change from stop-and-go motion to crowd turbulence.

Combing the extreme densities with low densities is researched even less until now. Only the article by Narain et al. [23] actually gives any consideration on this subject. Yet, this is an important problem. Extreme density scenarios often neglect to describe how agents come to join the high-density areas. Both low and extreme densities often occur together. An evacuation scenario already gives rise to this. At narrow passageways different paths might join. The flow consequently changes from a low-density to a high-density flow. The same can be said for a large open area and a set of stairs leading to the next level. The peak arrival at a train station is a visual example of this. We have extremely dense platforms once a train arrives. The main hall is already less dense as there is more space. And the exit-ways have even lower density as people disperse over several exits.

2.4 Combining both low and high densities

In this thesis we will extend current knowledge on crowd simulations by combining low and very high densities in one model. Up to our knowledge, the paper by Narain et al. [23] is the only paper that discusses this subject. In their paper they discuss the interpolation between agents moving on their own accord and the flow of the crowd. Agents can be forced by the crowd flow to move in a certain direction. Crowd flow in their paper is based on a global tendency of the crowd not to enter dense areas. Crowds are in this way perceived as a liquid. Flows will adhere to the properties of liquids. In contrast to their paper, we will not define a flow of agents as the liquid properties of the crowd. We will base our flow behaviour on the tendency of humans to follow other humans. This herding behaviour [57] in humans can most closely be related to Reynolds boids algorithm [37]. We will assume that humans have a tendency to follow other humans depending on the local density. This behaviour can be most clearly noted in large, crowded spaces such as railway stations. A recent paper by Lemercier et al. [58] backs up this assumption.

Definition 2 (Herding) *"The alignment of thoughts or behaviours of individuals in a group (herd) through local interactions rather than centralized coorination." [57]*

By basing flow behaviour on a local tendency rather than a global property, we can also handle crossing flows. There is no need to define a collective direction for each area of the environment. Agents belonging to the same group can make different choices while in the same area. Thus, two sets of agents can cross, though they belong to the same group.

We will interpolate between an agent's regular individual behaviour and its behaviour when faced with large crowds. This way, we will achieve an algorithm that can handle all densities. Agents will display realistic, believable movement for all possible densities and will have private goals to pursue. Agents dynamically join or leave streams (local flows), which in turn can cross each other if needed. This has not been shown before. All algorithms up-to-date can either handle highly dense crowds with group goals, or lower density crowds.

A more extensive discussion of the goals of this thesis is given in Part 1.3.

3 Preliminaries

Our research will extend upon previously performed research. In this chapter we will define the main concepts upon which we build. Readers should be familiar with these concepts before reading the rest of this thesis.

First we will describe the Explicit Corridor Map as proposed by Geraerts [28]. In the next section we will discuss the Indicative Route Method by Karamouzas et al. [5]. The last section will explain the vision-based avoidance algorithm as proposed by Moussaïd et al. [20].

3.1 Explicit Corridor Map

The Explicit Corridor Map (ECM) [28] is a technique to describe the free space in an environment. The free space is the set of areas that is not covered by a static obstacle. The ECM is a storage efficient data-structure describing the free space. It explicitly describes the entire two-dimensional free space. It is based on the concept of *Medial Axis*, with is closely related to *Voronoi Diagrams*. In Section 3.1.1 we will first discuss Voronoi Diagrams. The ECM will be discussed in Section 3.1.2. Lastly, in Section 3.1.3 we will explain how this concept can be used to extract routes for groups of agents.

3.1.1 (Generalized) Voronoi Diagrams

Voronoi Diagrams describe the distance relations within a set of points in a two-dimensional space. These diagrams have many applications within the field of mathematics and computer sciences. Common examples are the computation of the set of nearest neighbours and pattern recognition. Voronoi Diagrams are also referred to as *Dirichlet tessellations* [59].

In the Voronoi Diagram we have a space X and a distance measure d. This distance measure can be any kind of distance function. Often the Euclidean or Manhattan distance function is used.

In a Voronoi Diagram a set S consisting of n points is given. These point are called *sites*. Each site $s \in S$ is part of the space X. For each of the sites s, the Voronoi Diagram defines a subspace $X_s \in X$. All points $x \in X_s$ in this subspace are closer to s than to any other site s'. Within this subspace s is said to have *dominance* over all other sites s'.

$$\begin{aligned} Dominance_{s,s'} &= \{x \in \mathbb{R}^2 \,|\, d_{x,s} \leq d_{x,s'}\}, \\ \text{where, } d_{x,s} &= \text{ the distance between } x \text{ and } s \text{ using distance measure } d \end{aligned}$$

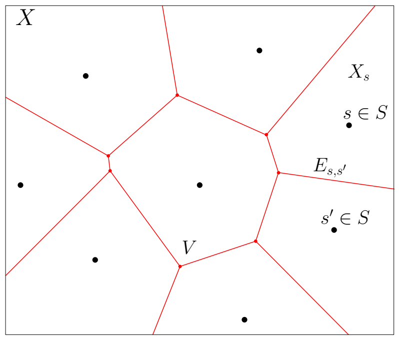


Figure 6: An example of a Voronoi Diagram. Sites are represented as black dots. Voronoi Edges are represented as red lines. Voronoi Vertices are red dots.

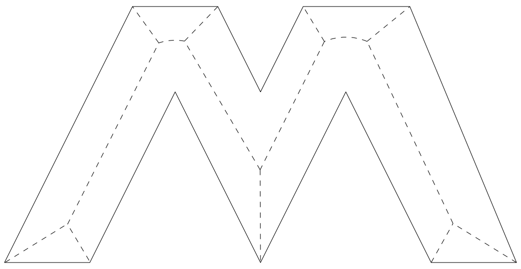


Figure 7: The medial axis of a polygon. The dashed lines describe the medial axis of the polygon.

Therefore, $X_s = \bigcap_{s' \in S - \{s\}} Dominance_{s,s'}$. These subspaces are referred to as Voronoi regions or Voronoi cells.

The set of sites S also define a set of points that are equidistant from two or more sites. For these points, no single site can be identified as the closest site. These sets of points are called Voronoi edges or Voronoi vertices. These points are interesting for our purposes. A point is part of a Voronoi edge ($E_{s,s'}$) if the closest two sites to it are equidistant from it.

$$x \in E_{s,s'} = \{x \in X \,|\, \forall s'' \in S,\, d_{x,s} = d_{x,s'} \wedge d_{x,s} \leq d_{x,s''}\}$$

Clearly this also implies that a Voronoi edge is on the boundary of two Voronoi cells. A point is a Voronoi Vertex (V) if there are three or more sites that are closest to it.

It can be proven that in a Voronoi Diagram the number of edges does not exceed $3n - 6$ and the number of vertices does not exceed $2n - 4$. The Voronoi Diagram is linear in the size of the number of sites. Therefore, the Voronoi Diagram is an efficient way to store information about an environment in 2D.

The notion of a Voronoi Diagram can be extended if we let the set of sites differ from just single points. In the *Generalized Voronoi Diagram* we allow the sites to have more complex shapes. Instead of consisting of a single point, sites can now consist of a set of points. In most common cases this will imply that sites can also consist of polygons, disks or lines. The distance measure is now defined as the smallest distance between a point x and some point y from site s. So $d_{x,s} = d_{x,y}$ where $y \in s \,|\, \forall y' \in s : d_{s,y} \leq d_{s,y'}$. Generalized Voronoi Diagrams (GVD) can be used to describe the connectivity in an environment. If an environment is fully connected, then the GVD will be so to. The GVD allows us to efficiently store the connectivity of any environment.

A lot of different extensions to the Voronoi Diagram are dubbed Generalized Voronoi Diagram. To avoid any confusion, we will end with the concept of medial axis. The medial axis is a subset of the points that are equally distant from at least two different points on the edges of one or more obstacle-polygons. All points on an edge meeting a reflex vertex are excluded. This concept is closely related to the GVD. We can assume the free space to be a polygon limited by the obstacles. The medial axis of this polygon equals the GVD minus all edges meeting a reflex vertex (Figure 7).

An extensive discussion about Voronoi Diagrams and their uses is given in [59]. The interested reader might refer to this paper for more information.

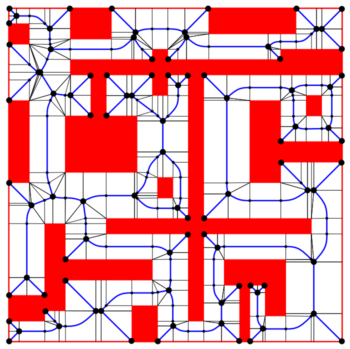


Figure 8: An example of the Explicit Corridor Map. Obstacles are represented in red. The medial axis are drawn in blue. Medial axis vertices are displayed as black dots. The smaller, black lines refer to the closest points for all event points (the vertices with degree two) and medial axis vertices.
(Figure by Geraerts [60])

3.1.2 The Explicit Corridor Map

Definition 3 (Explicit Corridor Map) *The Explicit Corridor Map is a Generalized Voronoi Diagram G=(V,E), where V and E are its Voronoi vertices and Voronoi edges. The edges are annotated with event points together with their closest points to obstacles. The set of event points is a discrete set of points located on the edges E. We define a parameterized function $B(t)$ of the edge describing the normals of the nearest obstacles in the respective closest points. A point p is defined to be an event point if the value of the function $B(t)$ start of stops to change in point p. [28]*

The Explicit Corridor Map (ECM) (Figure 8) is based on the concept of the Generalized Voronoi Diagram (or more precisely, the medial axis). It allows for efficiently storing the topology of the two-dimensional free space. The representation of a path in the ECM is one dimensional. Added annotations allow it to describe the entire free space. To enable this more points on the medial axis are designated as vertices. Agents can extract an explicit corridor, describing the two-dimensional space along a route. The ECM assumes all obstacles to be convex polygons, lines or points. Non-convex polygons are converted into convex ones.

The ECM describes the free space as related to the medial axis. As noted before, the medial axis describes the connectivity in the environment. Informally, it is the set of points located at the maximal distance to any nearby object. These two properties make it ideal to describe the free space. Every point in the free space is connected through a simple path to the medial axis. For every point on the medial axis, all nearby free points lie within a disc with known radius.

However, edges of the medial axis are not required to have a basic mathematical shape. This makes naively storing them inefficient. An edge consist of a concatenation of one or more simple curves. These curves are either straight line segments or of uniform arcs. Each of these curves can be stored efficiently if we know the end points. The ECM computes these end points and stores them as *event points*. Only near vertices of obstacles can the mathematical shape of the medial axis change.

For every vertex v_i of an obstacle we can compute the two normals (n_i,n_{i+1}) of its connected edges. We define two half-lines L_i and L_{i+1} in the directions of the normals starting in the vertex v_i. The nearest intersection for each half-line with the medial axis defines an event point.

Thus, event points, in combination with existent vertices, define the places where a change in path can occur. At these positions the ECM is annotated with the *closest points* on neighbouring

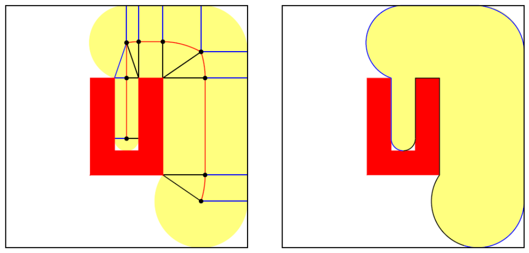


Figure 9: An example of an explicit corridor. a) The medial axis and closest point references are shown. b) The explicit corridor.
(Figures from [28] by Geraerts.)

obstacles. Annotations at these positions is sufficient to comprehensively describe the entire neighbouring space.

Straight-line segments along the medial axis can only be caused as a bisector between two opposing obstacles. If we know the closest points for both ends of the line segment we can interpolate all closest points.[5]

Arcs are by definition caused by a single point and another obstacle. This point can either be the corner of another obstacle or a point-obstacle. In both cases the arc is centred around the point. Thus, if we know the closest points, we also uniquely define the arc.

In conclusion, for the ECM we need to store the event points, the medial axis vertices and their respective closest points. The Voronoi Diagram is a space-efficient representation of the environment (see Section 3.1.1). The ECM adds at most $2n$ vertices for the event points. The total number of vertices is at most $4n - 4$, or $O(n)$. Thus the ECM is a storage-efficient representation of the free space.

3.1.3 Extracting an explicit corridor

The ECM allows an agent to quickly retrieve an explicit corridor describing the free space along a route. For all points in the free space we can easily determine the closest point on the medial axis. This retraction is always possible and the connecting line is guaranteed to remain inside the corridor. Both the goal and the agent's position can be retracted onto the medial axis. Subsequently, a search algorithm can be applied to determine a route towards the goal. Information about the medial axis and the closest points give a full description of the space.

Agents can query the ECM to retrieve an explicit corridor (see Figure 9). This corridor describes the two-dimensional space along a selected route towards its goal. The explicit corridor can quickly be retrieved from the ECM. When an agent queries the ECM, both its position and the goal position are retracted onto the medial axis. These first sub-paths are guaranteed to be safe. Around any point along the medial axis a circle can be drawn touching its closest obstacles. Any point in the free space, by definition, must be encompassed by a circle from some point on the medial axis. As no obstacles are present within this circle, a safe path is guaranteed towards the medial axis.

Once these retractions are determined, a search algorithm can be applied on the medial axis graph. Usually either the A*-algorithm or Dijkstra's algorithm is applied. These algorithms will compute the shortest global route between both retraction points. The concatenation of all three routes results in a full route towards the goal position. All vertices along the route contain information about the closest obstacle points. This allows us to construct the complete corridor efficiently, allowing the agent full freedom.

[5]Note that this definition includes the possibility that both obstacles are points.

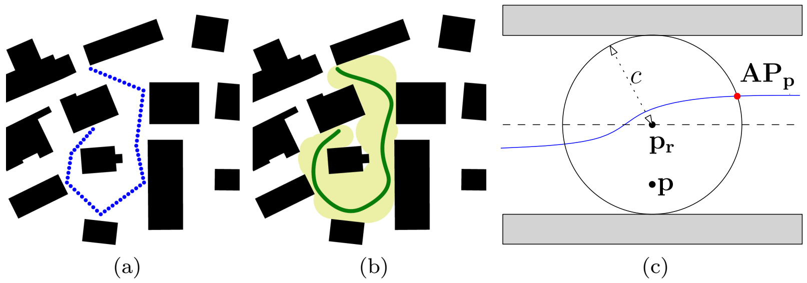


Figure 10: The Indicative Route Method. a) The original Indicative Route. b) The eventual path. c) The respective clearance disc, retraction point $\mathbf{p_r}$ and attraction point $\mathbf{AP_p}$ for an agent at position $\mathbf{p}$.
(Figures a *and* b *from [5] by Karamouzas et al.)*

If agents should keep a minimum distance to obstacles, the corridor can be shrunk. As agents are required to move inside the corridor, this will force them to keep a minimum distance[6]. Due to the event points this can be computed relatively easily. For more information we refer the reader to [28].

3.2 Indicative Route Method

In the paper by Karamouzas et al. [5] a method is proposed to make agents move more fluently. Agents are steered based on an attraction point that slides along a predefined route. This *indicative route* can be automatically generated or stated by the user. The method naturally integrates with the ECM as discussed in Section 3.1. A combination of forces keeps the agent from colliding with obstacles and other agents.

When agents are required to move along an exact route they are heavily constrained. This can cause them to move in an unrealistic way. In the Indicative Route Method (IRM) agents are no longer required to follow a route exactly. Instead, a rough indication of the preferred route is given. This route gives an estimation of the agent's exact path. When necessary, agents are allowed to make deviations from this path. Because agents are more able to participate with the environment, movement is more fluent. The IRM can easily be incorporated with the ECM by using the medial axis as an indicative route. It can, however, be applied to any indicative route. Figure 10a and 10b show an example of an Indicative Route and the eventually travelled path.

In the IRM an *attraction point* is defined that attracts the agent (Figure 10c). This point slides along the preferred route and continually causes the agent to move towards its goal. The attraction point is chosen such that there are never any obstacles between it and the agent's position. It will always lie ahead of the agent, making it move towards its goal. The combination of both factors assures that no local minima will occur, and, hence, the goal will be reached.

To compute the attraction point, the agent's position $\mathbf{p}$ is retracted onto the medial axis $\mathbf{p_r}$. For each point on the medial axis the distance (c) to the nearest obstacle is known. This information can be extracted from the Corridor Map. As $\mathbf{p_r}$ is on the medial axis, there exists an empty disc centred on $\mathbf{p_r}$ with radius c. This *clearance disc* is free of obstacles[7]. As the clearance disc is situated on the medial axis, it is centred in the middle of the corridor. Any point on the medial axis has at least two equally distant closest neighbours. Thus the clearance

[6]Agents have no other knowledge of the environment except the borders of the corridor. Therefore, they can never safely leave a corridor.

[7]Please note that these are obstacles due to the environment. Moving obstacles, such as other agents, might still be present.

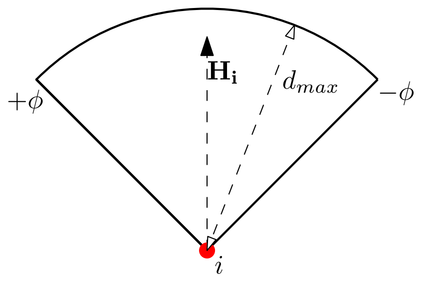


Figure 11: Graphical representation of the field of view of an agent i. The maximum view distance is d_{max}. The maximum angle of deviation is ϕ. The line of sight is $\mathbf{H_i}$.

circle is bound to touch at least two obstacles. This assures that the clearance disc intersects the entire width of the corridor. As any clearance disc intersects the entire corridor, it must intersect the indicative route as well. The attraction point is defined as the furthest point on the indicative route that intersects the boundary of the clearance disc.

To steer the agent, an attractive force is introduced. This force is directed towards the attraction point. It is integrated in the equation of motion to compute the new velocity. As the attraction point is always directed towards the goal, an agent will always try to move towards the goal position.

Besides this *steering force* different forces are added that apply on the agent. A *boundary force* is added to prevent the agent from escaping the corridor. There is also a *repulsive force* that forces agents away from obstacles. As is noted in the paper, these forces might be replaced by a more extensive collision-avoidance algorithm. In our thesis we will replace these forces by the vision-based avoidance algorithm as described in Section 3.3. In the original algorithm a random noise force is also introduced to create some path variation [5].

3.3 Vision-based avoidance algorithm

For collision avoidance algorithms different approaches have been explored. One of the more recent approaches is using a vision-based approach. The algorithm by Moussaïd et al. [20] is an example of this type of algorithm. The main assumption behind these algorithms is that humans perceive their world visually. Therefore it is assumed that the most realistic model involves visual information. Vision is the main information source used to control the motion of agents. In the first section we will define the basic assumptions taken in the algorithm by Moussaïd et al. In the next section a overview of the algorithm itself is given.

3.3.1 Assumptions and motivations

In the algorithm by Moussaïd et al. [20] several assumption are made. Each agent in the model is represented by a set of characteristics. Foremost each agent is characterized by its current position $\mathbf{x_i}$ and velocity $\mathbf{v_i}$. Besides these basic variables, agents are also endowed with a certain weight m_i. The agent is represented as a circle on the plane. The radius of this circle is defined as $r_i = \frac{m_i}{320}$. Thus an agent of average weight has an average radius. This common abstraction from reality allows for more efficient computations. Each agent also has a preferred walking speed v_i^0 and a destination point O_i.

As this avoidance algorithm is based on vision, each agent also has a given *field of view*. This field of view determines the part of the environment that this agent is able to perceive. It is a cone with origin at the agent's position $\mathbf{x_i}$. This cone is bounded by a maximal angle ϕ as related to the line of sight $\mathbf{H_i}$. It is also bounded at a maximal distance d_{max}. See Figure 11 for a visual depiction.

The field of view determines the neighbourhood of an agent. Only agents that are within this field will be taken care of in collision avoidance. The field of view also determines the maximum allowed change of direction per second that can be selected, which is equal to ϕ.

In the algorithm it is assumed that agents are continuously adapting their current walking behaviour to match their desired behaviour. There is a relaxation time τ of 0.5s. This implies that an agent is assumed to require 0.5s to reach any desired velocity. It also implies that an agent is required to keep a 'distance' of 0.5s to any static obstacle or agent. If necessary, the agent should slow down to maintain these 0.5s. If this is the case than an agent can always stop in time.

To process the information as provided by the virtual vision, two heuristics are introduced. One heuristic decides on the walking direction and the other determines speed.

- "A pedestrian chooses the direction α_{des} that allows the most direct path to destination point O_i, taking into account the presence of obstacles."
- "A pedestrian maintains a distance from the first obstacle in the chosen walking direction that ensures a time to collision of at least τ."

From these heuristics we can derive several properties. First of all, if possible, an agent prefers to move in the most straight-line towards its goal. Second of all, there is a trade-off between taking the most direct route and avoiding collisions. This trade-off can be seen in many different papers as well and is related to energy-expenditure. Agents try to spend as little energy as possible when trying to reach their goal [4, 61]. Thus it is most efficient to take the shortest route possible. However, as any change in velocity requires energy, so does a speed change. Taking a minor detour may be more energy-efficient than taking the shortest path. In the algorithm this speed change is related to the *distance to collision* $f(\alpha)$ for a given course α. The smaller this distance, the more likely that a speed change is required. Thus distance to collision has an inverse effect on the attractiveness of a route. The same can be said for the course α. The value of α is measured as the angle to the most direct path. The more an agent deviates from the direct path, the less attractive a path becomes.

For each direction a relative attractiveness can now be computed. The chosen direction α is determined through the minimization of the function[8]

$$d(\alpha) = \sqrt{d_{max}^2 + f(\alpha)^2 - 2 * d_{max} * f(\alpha) * \cos(\alpha_0 - \alpha)}, \qquad (1)$$

where,

$$\begin{aligned} d_{max} &= \text{the maximal view distance,} \\ f(\alpha) &= \text{the distance to the first collision for the tested direction,} \\ \alpha_0 &= \text{angle between } \mathbf{H_i} \text{ and the most direct direction,} \\ \alpha &= \text{angle between } \mathbf{H_i} \text{ and the tested direction} \end{aligned}$$

This function minimizes the distance from the first collision in direction α to the furthest visible point in the direction α_0 of the goal position.

This can be seen in Figure 12. We can compute the distance $d(\alpha)$ by a combination of basic trigonometry rules.

[8]We point out that the square root is not present in the original paper. Contact with the author, however, showed this as the original formula. We note that the square root, as well as d_{max}^2, can be omitted as $d(\alpha)$ is only used in a minimization.

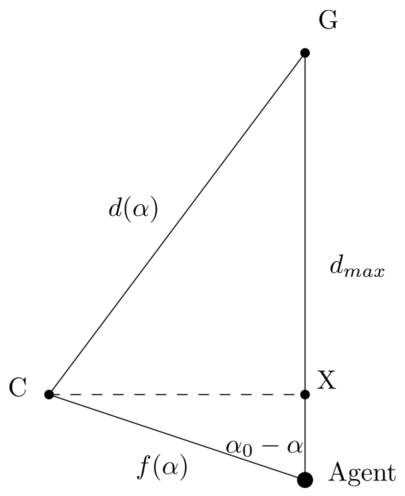


Figure 12: The value of $d(\alpha)$ represents a distance function to the desired point G. We assume C is the first point of collision in direction α. The vertex G is the furthest visible point in the direction α_0 of the goal position. The point X is the foot of the altitude CX. We can now deduce $d(\alpha)$ by computing the different sides.

$$d_{X,Agent} = f(\alpha) * cos(\alpha_0 - \alpha)$$
$$d_{X,G} = d_{max} - f(\alpha) * cos(\alpha_0 - \alpha)$$
$$d_{X,C} = \sqrt{f(\alpha)^2 - d^2_{X,Agent}}$$
$$d(\alpha) = \sqrt{d^2_{X,G} + d^2_{X,C}}$$

3.3.2 The algorithm

For ease of computing, the continuous set of angles is discretised, leaving a discrete set of options. During each simulation step, for each agent the algorithm computes the most energy-efficient choice of direction. This direction is a trade-off between the shortest path and the fastest path, thus maximizing energy-efficiency. Note that this is a heuristic. Other agents are assumed to remain on the same course indefinitely, whereas they are most likely not. This may cause the situation to change, causing the current deliberation to be invalidated. All algorithms that do not assume complete knowledge are by definition heuristic.

To compute the optimal direction we first need to know all local obstacles. The static scenery is known, but other agents might also block the way. All agents that are situated inside the field of view are taken into account. Then, for each possible direction, the distance to the first predicted collision is computed. The assumption is made that all neighbouring agents will continue on their current velocity. This allows us to predict where they will be in the upcoming time. For the current agent, it is assumed that it will travel at its preferred speed. A direction that is optimal at the preferred speed is likely also acceptable at a lower speed.

A naive way to compute all the collision distances would involve testing for each direction the collision times with all neighbouring agents and storing the smallest value. For each direction it should also be checked if a collision with the static scenery occurs. If no collision occurs for a given direction, a maximum value d_{max} is assumed. In the addenda we will discuss a less naive solution that improves computation times.

As agents dislike deviating too much from the direct path to the goal, but also seek an unobstructed walking direction [20], a trade-off has to be made. Each possible direction is scored by its distance to the first collision and its deviation from the direct path (see Equation

1). The direction with the best score is selected as the optimal direction. Having decided upon a direction, the optimal speed is determined. This is defined as the minimum of the preferred speed and the highest speed which will assure no collisions occur if all other agents remain stationary. The latter is computed by taking the distance to the first collision and the reaction-time τ. Thus, if collisions are unavoidable the agent will come to a full stop. The velocity obtained by combining the preferred speed and direction will be the desired velocity for an agent. The agent's position in the next timestep is computed by the basic formula from Physics $pos_{new} = pos_{old} + timeStep * v_{new}$.

The acceleration for the agent is the difference between this desired velocity and the current velocity divided by τ. This acceleration can subsequently be used to compute the new velocity. The acceleration is not bounded in the algorithm. So it can maximally be twice the preferred speed with the reaction time of τ seconds. With the base value for τ this becomes four times the preferred speed per second square.

The algorithm so far will make sure agents avoid each other and try to take the most energy-efficient route towards their goal. In case of high-density, however, agents can still come into collision with each other or with the environment. Therefore extra forces are added that push agents away from the colliding object. These forces are at all times equal to zero, unless the agent is in a collision. By adding the sum of all forces to the acceleration term described before, agents will prevent collisions, but also resolve them. It is important to note that these forces are larger than zero only if there is actually a collision, in contrast to the social-force model [1].

The vision-based algorithm is a realistic model of how humans perceive the environment. Agents try to move along an energy-efficient path. Through the use of the heuristics it is a computationally efficient approach. The algorithm can resolve collisions correctly at all densities, which we require. Due to the lack of inter-agent coordination, however, the algorithm has trouble with high-density situations.

In high-density scenarios agents should be willing to sacrifice temporary personal advantages in favour of the flow of the crowd. For example, agents should not leave a stream because there is a temporary opening in the opposing stream. As agents are, however, only able to make individualistic decisions, they will always take egocentric decisions. This leads to the sacrifice of lane formation and causes deadlocks and suboptimal paths.

3.3.3 Additions to the basic algorithm

Though the original algorithm by Moussaïd performs very well, we have made a number of minor changes before running the experiments. These improvements are mainly focused on correctly maintaining inter-agent distance.

First of all we have introduced a minimum speed for all agents. If agents are forced to move at a lower speed they will come to a full stop. This improves stop-and-go motions in crowds. We have set the lower speed at a value of 0.06m/s.

Secondly, we have changed the desired speed that agents select. In the original algorithm agents select their speed based on the current positions of all visible neighbouring agents. This helps agents keep a correct distance to predecessors that are moving in the same direction. Even if a predecessor instantly stops the agent can come to a full stop in time as well.

However, if neighbouring agents move in an opposing direction this is troublesome. By only taking into account the current position of these agents and not their current speed a too high speed is selected. We can not guarantee that the agent will be able to stop in time. As this behaviour is displayed by both agents, problems are even further amplified.

To solve this problem we select the speed for an agent based on both the current and anticipated future situation. We compute the distance to collision assuming agents remain stationary, as well as the distance to collision assuming agents keep their current velocity. The latter is already computed by the algorithm so this does not require any extra computations.

To avoid collisions in all scenarios, we select the smallest collision distance of both.

Lastly, we keep track of the distance from the agent to its current attraction point. If this distance is small then we lower the maximum speed that the agent can assume. This is required to make sure that an agent can turn fast enough if its attraction point is very near. This is most often the case if the agent nears its goal position.

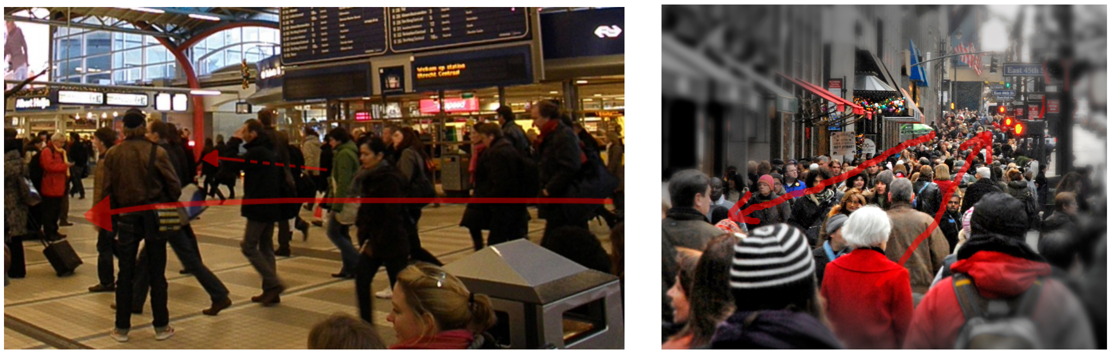


Figure 13: Streams at medium density flowing across a train station and at high density enforcing and increasing lane formation.

4 Streams

Humans have a tendency to follow others [58]. This herding behaviour seems counterintuitive to our individualistic nature. But in areas of high-density this behaviour is actually rather common. Once density is sufficiently high, agents require a common strategy in order to reach their goals efficiently. This common strategy will allow information to be passed back through the crowd. We can relate this to a form of herding-behaviour[9]. In this chapter we will propose the use of a stream approach to mimic this high-density behaviour. Instead of being based on the fluid-properties, however, we will base this flow on the *perceived local flow* of a crowd.

In Section 4.1, we will formally define the concept of streams and clarify its motivation. In Section 4.2, we will discuss a local, vision-based implementation. We would like to stress that the concept of streams is unrelated to agent goals. Streams can point in contra-productive directions and are not guaranteed to be free of local minima. This is not required for our approach either. As we interpolate between the stream and individualistic behaviour this will not cause a problem. If the stream causes an agent to move in a direction which deviates too much, individualistic behaviour will take over once more. This same switching of behaviour causes agents to leave local minima of streams. As soon as an agent moves past this point, the direction of the stream is guaranteed to be counterproductive and the agent will make an individualist choice once more. The interpolation meta-strategies will be discussed in Chapter 5.

4.1 Definition of a Stream

Before we define the local stream approach, let us first sketch an example to clarify the concept. Picture a train station during rush hour. There is a large hallway that commuters are traversing to reach their platforms. Everybody is trying to reach their train in time and has a given platform to reach. For all agents time and energy efficiency are a crucial factor for determining movement [4, 61]. Therefore, we would expect all commuters to take the shortest possible route towards their platform whilst avoiding other commuters. Maybe suprisingly, this is often not the case. Local 'streams' of commuters 'flow' across the hallway. Only at the last moment in time, people leave the stream to go to their platform.

Humans have a tendency to follow the *streams*. And once they follow a certain stream, they are more likely to stay with it. Only when the direction of the stream deviates too much, will they contemplate leaving it. Even though a straight-line path towards the goal existed, a slight

[9]A definition of herding is supplied in Section 2.4.

detour is taken to move along with the stream. We will consequently stick with the term *streams* to describe this phenomenon. This emphasizes the difference with the goal-based, global flow fields as described in other papers [22, 24, 25].

Definition 4 (Stream) *The local stream S_A is the* perceived *local flow-velocity* $\mathbf{v}$ *in a point* $\mathbf{p}$ *as perceived by an agent A. This flow-velocity is an interpolation*[10] *of the* perceived velocities *of all agents A_i within a Euclidean distance d of agent A. Different interpolation-scheme's may be used to represent different perceptions.*

Definition 5 (Perceived velocity) *The perceived velocity* $\mathbf{p_{A,N}}$ *of an agent N by an agent A is the contribution of agent N to the local stream as perceived by agent A. The perceived velocity is a composition of agent N's velocity and position based on the alignment and follow strategy of agent A. A mathematical definition for the vision-based perceived velocity will be supplied in Section 4.2.1.*

It is important to notice that the stream-concept is a *local* concept. Regular flow fields describe a global behaviour. These flow approaches are designed as a global requirement field. They either lead the agents towards a specific goal [22] or reroute agents around extremely dense areas [23]. Streams, in contrast, are a local 'desirable movement'-vector based on the movement of neighbouring agents. Also note, depending on the implementation, that not all agents within the distance d might be taken into account. An example of this will be given in Section 4.2.

The local movement of other agents provides insight in the most efficient direction of movement. Information is implicitly being passed by the agents in front about the situation ahead. Even if we can not see this situation yet, information is still being conveyed on the best course of action. The more congested an area becomes, the more important it becomes to use this information. In extreme densities agents should cooperate highly to allow any movement.

The denser it gets, the higher the social pressure is to comply to this group motion. It is also in the agent's advantage to move along the stream. By using a group strategy, a more energy-efficient path can be achieved than by taking an individualistic strategy. As density increases, the disadvantage of taking a personally less optimal strategy decreases. Thus agents will be more likely to switch strategy and follow the group.

There are several advantages to the stream-based approach as opposed to the use of flow fields.

- Agents can have individual goals.
- We can easily integrate different agent behaviours. There is no limit to the number of different groups. Theoretically, each agent could have its own settings or preferences.
- We can allow for agent behaviour dependent on the local conditions. For example, agents on the outside of a stream can display different behaviours than those within even though they have the same goal.
- Crossing flows are implicitly supported.

These are the main drawbacks of flow fields and are a limiting assumption when modelling reality.

The main disadvantage of the stream-based approach as compared to flow fields is its computational complexity. Flow fields require one computational sequence to compute the movement for a group of agents. The stream-based approach requires computations for each agent. This is a direct consequence of the desire for individual goals.

Streams are not a navigation approach for agents. They should be seen as a social pressure that an agent experiences when in a specific location. Social pressure is not only a social

[10]Interpolation is defined as the weighted average where the sum of the weights equals one.

enforcement by the environment on an agent. This pressure is partly an internal pressure as well. It is in an agent's advantage to comply to this pressure. Local streams allow an agent to easily move in the direction of the stream. The protection of the group allows an agent to spend less time on avoiding collisions. This collision avoidance ('paying attention to the environment') is also an energy demanding activity. Thus following the stream is an energy-efficient choice of path. Agents try to follow a least-effort path [4, 61] and are therefore likely to follow the stream. Of course multiple factors influence this behaviour in real life. For now we will assume a full desire to follow the stream. A discussion of when to adhere to this strategy and when to deviate is deferred to Chapter 5.

4.2 Local vision-based streams

The stream-based approach we have designed is a local, vision-based approach. From our preliminary experiments it turned out that the locality of this vision best fits the stream-approach. Agents visually perceive the agents ahead of them. We will try to distil from these neighbours what the current stream is at the point of measurement.

Agents are assumed to have a given *field of view*. This field of view is a cone stretching out from the agent's current position. The cone is centred on the vector pointing in the direction of movement. It reaches out in both directions up to a maximum viewing angle. The cone is bound by a maximum distance at which the agent can perceive the environment. Any agents that are within this field of view will be defined as local neighbours for the stream definition. This cone will be used as well to determine the local density, but up to a smaller distance. The main problem we are faced with is how to distil the stream at the agent's current location from agents that are ahead of it. We have studied the movement of crowds at several train stations and at a major Dutch event (SAIL). From this we conclude that an interpolation between several local variables is required. We will discuss these variables and their motivation below. In Section 4.2.1 we will discuss how a single neighbour is perceived. In Section 4.2.2, we will define how the local stream can be extracted from all neighbours within the field of view. Section 4.2.3 will discuss several properties of the stream approach as well as some of its limitations.

4.2.1 Perceiving a single neighbour

To determine the local stream-velocity, we first need to determine the perceived velocity of a single agent. This perceived velocity represents the contribution to the local stream a single neighbouring agent N makes. It is not necessarily equal to the movement velocity of agent N. A trailing agent A will be more inclined to move towards agent N if it is far way, as opposed to moving along. The perceived velocity is constructed from the velocity of movement and the relative direction of agent N as opposed to agent A. The combination of both velocities generates an approximation of the local stream velocity. That is, the velocity that agent A should assume to move along in the stream.

The *perceived velocity* of an agent N as perceived by agent A is dependent on its movement and its relative position. These vectors are interpolated based on the distance between agents and the local density. Depending on the distance an agent requires a different strategy to move along the stream. Agents react differently at different densities. At higher densities a more compact formation is desirable. The resulting velocity is the agent's contribution to the local stream.

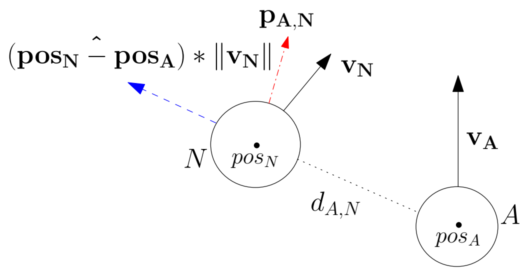


Figure 14: The perceived velocity of an agent N as perceived by an agent A. In black are the actual velocities of both agents. The perceived velocity (in red) is composed of two components. The actual velocity of the agent N and the agent's scaled relative position, represented in blue. The perceived velocity is specific for a pair of agents A and N.

Definition 6 (Perceived velocity (visual)) *The perceived velocity $\mathbf{p_{A,N}}$ of an agent N by an agent A is defined as (Figure 14):*

$$\mathbf{p_{A,N}} = f_{A,N} * (\mathbf{pos_N} \hat{-} \mathbf{pos_A}) * \|\mathbf{v_N}\| + (1 - f_{A,N}) * \mathbf{v_N}$$

where

$$\begin{aligned}
(\mathbf{pos_N} \hat{-} \mathbf{pos_A}) &= \textit{the normalized vector } (\mathbf{pos_N} - \mathbf{pos_A});\\
\mathbf{pos_N} &= \textit{the current position of agent } N;\\
\mathbf{pos_A} &= \textit{the current position of agent } A;\\
\mathbf{v_N} &= \textit{the velocity of agent } N;\\
f_{A,N} &= \textit{the interpolation factor for agent } N \textit{ as perceived by agent } A;
\end{aligned}$$

and $f_{A,N} = D_A * d'_{A,N}$ where,

$$D_A = \textit{the local density at agent } A \ (0 \leq D_A \leq 1);$$

$$d'_{A,N} = \textit{the relative distance } (0 \leq d'_{A,N} \leq 1) \textit{ between agent } A \textit{ and agent } N. = \frac{d_{A,N}}{d_{max}}.$$

Let us first define this more extensively. For now we will assume that the neighbourhood of an agent A is solely comprised of a single agent N. When moving along the stream, agent A needs to execute two different strategies. These strategies will extrapolate the stream, as perceived in front of the agent, to its current position. The first strategy will cause it to attempt to move to the position at which it currently perceives agent N. We will refer to this as the *follow* strategy. Agent A will try to move along a vector v defined by the respective positions ($\mathbf{v} = \mathbf{pos_N} - \mathbf{pos_A}$). However, as soon as two agents get closer to each other, problems start occurring. As can be seen in Figure 15a, agents get attracted towards each other at close range. This causes them to actively seek out collisions instead of avoiding them.

For nearby agents, agent A should pick a different strategy and move along with its neighbours ($\mathbf{v_A} = \mathbf{v_N}$). This strategy is an *alignment* strategy. This side-by-side movement also occurs when agents move together as a group. By applying this strategy, agents will line out. However, it causes problems if agent N is ahead of agent A (see Figure 15b).

Therefore an interpolation between both strategies is required. To interpolate we make use of a distance factor ($0 \leq d'_{A,N} \leq 1$). If a neighbouring agent is on the edge of the view-distance we will only apply the follow strategy to determine the local stream-velocity. For an infinitely close neighbour the alignment strategy will be applied. All neighbours in between will cause an interpolation between these velocities, dependent on the distance to agent A. Note that this

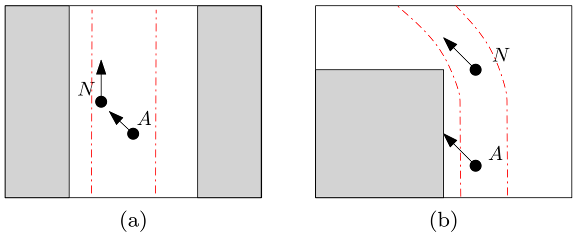


Figure 15: Problems occuring when trying to define the local stream for an agent A. The actual stream is displayed as red-dashed lines. (a) Tailing neighbouring agents leads to unnecessary collisions. (b) Following the current velocities of neighbours is also troublesome.

combination of strategies will make agents pick a more realistic stream direction. There are still cases where this does not lead to the desired solution. Agents can still clip the corners of obstacles. However, this behaviour is far less extreme and can easily be corrected by a collision detection algorithm.

If agents are nearly stationary, their effect on the local stream lowers. This helps upcoming agents avoid stationary areas. However, it also breaks down stream coordination in highly congested scenarios where general speed is low. To prevent this we have introduced a minimum threshold on the applied velocity. If $\|\mathbf{v_N}\| < 0.3$, we replace $\mathbf{v_N}$ by $0.3 * \hat{\mathbf{v_N}}$. Thus, slow moving agents always have a minimum contribution. For stationary agents we use its current line of sight instead of its velocity, giving rise to $0.3 * \mathbf{H_N}$. The value of 0.3 was determined through preliminary experiments.

Depending on the density, agents need to display a different degree of coordination. At high densities we want to obtain a more compact formation of agents, whereas, at lower densities, a greater spread along the corridor is required. This lane formation is a known phenomenon in real crowds [1, 4, 22]. At high densities we want agents to be more inclined to move towards other agents. In other words, the location-component ($\mathbf{pos_N} - \mathbf{pos_A}$) of the perceived velocity is more important than the direction-component. At low-density however, we want agents to use a larger portion of the corridor. In these cases the direction-component ($\mathbf{v_N}$) is more important than the location-component. In order to mimic this behaviour, we have scaled the distance-factor by the local density ($0 \leq D_A \leq 1$). By scaling the distance-factor by the density, we can prove that as density increases the stream start to narrow more strongly. This is required to maintain correct lane formation in higher densities.

In order to prove that streams contract more strongly as density increases we make the following assumptions.

- We assume an arbitrary agent x which is part of a stream S with velocity $\mathbf{v_S} > 0$.
- All agents in the stream move with a velocity equal to the stream.
- Agent x is on the outside of a stream.
- All agents within stream A move in the direction of the stream ($\mathbf{v_S}$).

Theorem: *At high density, streams will become narrower, thus allowing for better lane formation*

Proof: We prove that agent x will be pulled more towards the center of the stream, as density increases. To do so we will compare the most extreme density values possible, namely 0 and 1.

Let us first assume, that the local density is 0. Note that this is an unrealistic value as at this density no stream will be present. Still it will exemplify our proof best. As $D_x = 0$ it must also be the case that $\forall$ agent i in the field of view (FoV_x): $f_{x,i} = D_x * d_{x,i} = 0$. According to Definition 6, we know that the perceived velocity $\forall$ agent $i \in FoV_x$ by agent x equals $\mathbf{p_{x,i}} = f_{x,i} * (\widehat{\mathbf{pos_i} - \mathbf{pos_x}}) * \|\mathbf{v_i}\| + (1 - f_{x,i}) * \mathbf{v_i}$. From the previous two statements we can deduce that $\mathbf{p_{x,i}} = \mathbf{v_i}$. So we know that the perceived velocity of all neighbouring agents i equals their actual velocity. As we assumed that all agents in the stream move with a velocity equal to the stream, we know that $\forall$ agent $i \in FoV_x$: $\mathbf{v_i} = \mathbf{v_S}$. Thus the weighted average of the perceived velocities $\mathbf{p_{x,i}} = \mathbf{v_i} = \mathbf{v_S}$ must equal $\mathbf{v_S}$. This means that the outer agent x will not move towards the center of the stream. As agent x was an arbitrary agent on the outside of the stream, this holds for all outer agents. So the stream will not become narrower.

For the other extreme case, we will assume a local density of 1. As $D_x = 1$, we know that $f_{x,i} = D_x * d_{x,i} = d_{x,i}$. According to Definition 6, we know that $\forall$ agent $i \in FoV_x$ the perceived velocity equals $\mathbf{p_{x,i}} = d_{x,i} * (\widehat{\mathbf{pos_i} - \mathbf{pos_x}}) * \|\mathbf{v_i}\| + (1 - d_{x,i}) * \mathbf{v_i}$. As agent x is on the outside of the stream, it must be that case that $\forall$ agent $i \in FoV_x$, $(\widehat{\mathbf{pos_i} - \mathbf{pos_x}})$ is directed towards the center of the stream. We can now distinguish two different cases. In the first case, there is an agent $i \in FoV_x$ that is located at a distance $d_{x,i} > 0$. In the other case all agents within the FoV are located at a distance 0.

Let us handle the first case and assume $\exists i \in FoV_x : d_{x,i} > 0$. As $d_{x,i} > 0$, it must be that $\|(\widehat{\mathbf{pos_i} - \mathbf{pos_x}})\| = d_{x,i} > 0$. Thus, the vector $(\widehat{\mathbf{pos_i} - \mathbf{pos_x}})$ is a non-null vector. Because $\|\mathbf{v_i}\| > 0$ as well, $d_{x,i} * (\widehat{\mathbf{pos_i} - \mathbf{pos_x}}) * \|\mathbf{v_i}\|$ is a non-null vector directed towards the center of the stream. As $\mathbf{v_i}$ equals the velocity of the stream the perceived velocity $\mathbf{p_{x,i}}$ is guaranteed to point towards the center of the stream. We have shown that at a density of 0 no movement towards the center of the stream occurs. We have also shown that at a density of 1, agents move towards the center of the stream. By using the value of $f_{x,i}$ it is trivial to show that this behaviour increases in strength as density increases. Q.E.D.

For the second case we will assume that there is no neighbouring agent who is located at a distance $d_{x,i}$ larger than 0. In other words,$\nexists i \in FoV : d_{x,i} > 0$. So $\forall$ agent $i \in FoV_x : d_{x,i} \leq 0$. As $d_{x,i} \geq 0$, it must be that $\forall$ agent $i \in FoV_x : d_{x,i} = 0$. This translates to the case, where all neighbouring agents are in contact with agent x. This is a degenerate case, which we will ignore. The resulting $\mathbf{p_{x,i}} = \mathbf{v_i}$ is a reasonable response. Q.E.D.

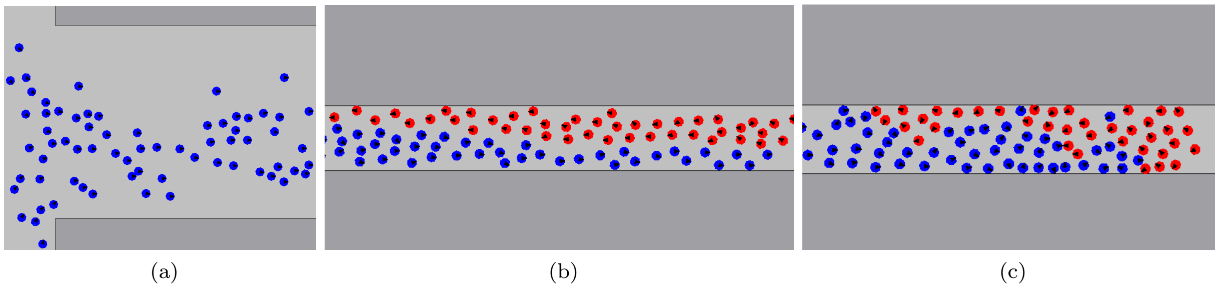


Figure 16: The amount of coordination required for a stream to form. When too much coordination takes place, agents stick together too extremely. There are, however, cases where this is required. (a) Too much coordination in low-density situations gives rise to a narrow stream, where a wide corridor is available. (b) In high-density situations this amount of coordination is required. (c) If coordination is lowered, lack of coordination causes havoc in high-density situations.

In Figure 16, a visual depiction of this density dependent problem is supplied. In low-density areas, agents have too much preference for following the stream. This results in a narrow stream of agents moving through a wide corridor (Figure 16a). This inefficient use of space is not realistic. At high densities, on the other hand, a high level of coordination in a group is required. Even more so if two opposing streams of agents meet in a relatively narrow corridor (Figure 16b and 16c).

Please note that this behaviour is also dependent on the collision-avoidance approach that is used. Some algorithms are inherently better capable of handling higher density situations. However, in our research we have not found a collision-avoidance algorithm that can handle these high density situations correctly. The are algorithms that can handle these density correctly, but only in uni-directional situations. Some reasons can be pointed out for this lack of algorithms. First of all, in dense crowds, agents are forced along by the flow. Other movement choices are not an option for an agent, even if space is available. Agents should adopt the required direction instead of keeping true to their personal goal. This, however, requires coordination which is lacking in current algorithms.

Second of all, at extreme densities, basing agent movement only on the local situation is unrealistic. Reasons can exist outside of the local area that have effect on the choice an agent should make. We may not be able to see the cause, but we should not neglect the information stored in the choices of other agents. Streams allow for the passing of this information.

An example of the above we look at two opposing groups of agents. The information that an opposing group of agents is nearby is implicitly passed to the rear. Agents start to avoid each other and this change in direction is noticed by the agents behind. This causes a cascade of direction changes. Thus, agents start to coordinate their movements even if no cause is yet within view range. The external coordination in the form of streams allows for correct handling of these situations. The resulting perceived velocity delivers the required behaviour at all densities.

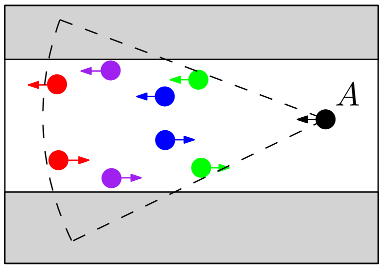


Figure 17: Averaging perceived velocities. A equally weighted average of the perceived velocities results in an undesirable stream-velocity. All equally coloured agents nullify each other. As a consequence agent A would perceive a stream-velocity equal to the null vector, while clearly a stream does exist.

4.2.2 Perceiving the local stream

The perceived velocity of an agent N will contribute to the stream-velocity of an agent A. For all neighbouring agents these perceived velocities need to be averaged. We will take a weighted average of the velocities as dependent on the direction. This is required to detect the correct stream in a crowd of moving agents.

As a simple example of the need to weight the different velocities we will examine a simple scenario (Figure 17). Suppose we have two oppositely moving groups of agents that are moving through a corridor. Intuitively we can state that there are two streams occurring in this corridor. One in either direction. However, in agent A's field of view an even number of agents are visible moving in either direction. These agents are pair-wise at equal distance from agent A and have opposing velocities. A naive sum of perceived vectors will compute an average stream-velocity for agent A equal to the null vector. This is clearly undesirable. We need to select the agents belonging to the correct stream.

To achieve this we take the weighted average of all perceived velocities. The more a neighbouring agent is moving in the same direction as the current agent, the more its perceived velocity is factored in. This represents the amount by which this perceived velocity is important for our current agent. This averaging scheme will also solve the crossing of two streams. If the angle between streams is large enough, agents will only *perceive* the stream in the direction that they are moving. If streams cross at a shallow angle, however, both streams will merge.

Note that it might be agent dependent to what degree perceived velocities are weighted. In Formula 2 this averaging scheme is presented. Depending on the angle between both agents' velocities $(\hat{\mathbf{v_A}} \cdot \hat{\mathbf{v_N}})$ a weight is attributed to the perceived velocity of agent N.

$$w_{A,B} = \alpha + \left(\frac{(\hat{\mathbf{v_A}} \cdot \hat{\mathbf{v_N}}) + 1}{2} \right)^{\beta} * (1 - \alpha) \tag{2}$$
$$\alpha \in [0,1]$$
$$\beta \in [0,\infty),$$

where

$$\hat{\mathbf{v_A}} = \text{unit velocity of the current agent } A (= \text{the direction of travel});$$
$$\hat{\mathbf{v_{A,N}}} = \text{unit velocity of neighbouring agent } N;$$
$$(\hat{\mathbf{v_A}} \cdot \overrightarrow{v_N}) = \text{dot product of } \hat{\mathbf{v_A}} \text{ and } \hat{\mathbf{v_N}}.$$

We insert two agent-based factors α and β. Variable α determines to what degree an agent is affected by perceived velocities, no matter what their relative direction is. The higher this value, the more an agent will be influenced by agents moving in a different direction. This value can be compared to the indecisiveness of an agent. At its base value of 0, an agent will attempt to follow the current stream. At the extreme value of 1 all options are open and an agent will follow the mostly represented direction. The value of variable β determines the extremity between options. The higher this value, the stronger the distinction between agents in our current stream and agents moving in different directions.

To further improve the selection of correct velocities, we insert an upper bound on the angle that the *actual* current velocities of two agents may differ. Through preliminary experiments we set this value at an angle of 90º.

We can now determine the local stream velocity by summing all the weighted perceived velocities. Not all neighbours within the view range need to be taken into account. Instead we take a subset of x agents. These x agents are the nearest x agents. An alternative is to take the first x agents with which a collision will occur. In our thesis we will take $x = 5$ agents. This is in agreement with the results in other papers on the required number of neighbours [2, 4]. Please note that taking into account more neighbours does not make the algorithm more precise. Streams are a local concept and should not be interpolated over a large area. There is a relation with the concept of nearness in the boids algorithm by Reynolds [37]. The movement of a flock or crowd only makes sense at a local scale. If we interpolate over too many agents, velocities become meaningless. Just as there is a steep drop off in the weighting of the boids model, agents should not be overly bothered by other agents that are relatively far away. We can now define the local stream as follows:

Definition 7 (Stream (visual)) *The local stream* $\mathbf{S_A}$ *for an agent* A *is defined as:*

$$\overrightarrow{S_A} = \frac{\sum\limits^{N \in visible} w_{A,N} * \mathbf{p_{A,N}}}{\sum\limits^{N \in visible} w_{A,N}},$$

where

$w_{A,N}$ = *the weighting factor for an agent* N *as perceived by agent* A*;*

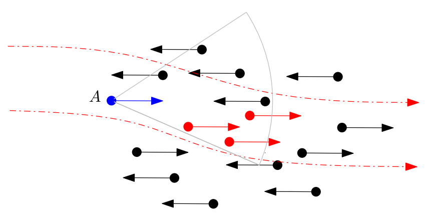
$\mathbf{p_{A,N}}$ = *the perceived velocity for agent* N *as perceived by agent* A*.*

The set visible *consist of the nearest* x *agents within agent* A*'s field of view.*

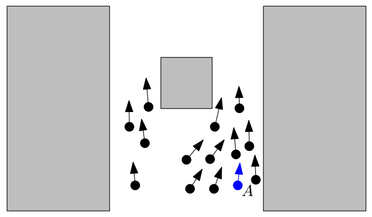
4.2.3 Properties

We will display some of the properties of streams in this section. There are still some cases in which we are not yet capable of determining the correct stream. These cases are often solved by the merging technique discussed in Chapter 5. Some cases can only be solved by the micro avoidance algorithm (see Section 3.3). These techniques will make sure that agents will keep displaying realistic behaviour. Optimally, however, we would eventually like to solve these cases as well. This might be interesting for future research.

Due to the setup of streams, agents are able to determine the location of streams even when in a large mass. As can be seen in Figure 18a, agents will automatically be influenced most by those agents that are important to it. When the blue agent A determines the local stream, it will detect seven agents within its field of view. The four black agents on the top left and the three red agents in the middle. As the red agents move in the same direction, their preferred velocity will be weighted most. Due to the setup of the perceived velocity, the local stream for agent A will point towards the red agents. Therefore it will fall in line and follow the red agents.



(a) Streams affecting lane-formation in two-way traffic.

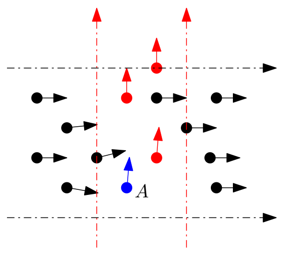


(b) Following the stream to allow for a better route.

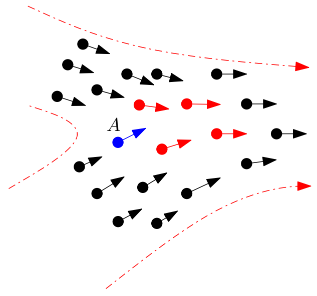
Figure 18: The effect of following streams in different situations. The agent A in question is drawn in blue. Important neighbours are represented in red. The stream is represented by red dash-dotted lines. In the Figure b, streams will force agent A to pass on the right of the obstacle, even though its planned route passed it on the left.

Streams will also help agents take a more logical route. Let us assume a situation as given in Figure 18b. Suppose agent A has planned its global route to pass the centre obstacle on the left. This would require it to move through the crowd of oppositely moving agents in the centre. As density is high, however, it will follow the local stream, thereby passing the obstacle on the right. Note that this will force a replanning of its global path. If no path is possible from this place onward, its new route will drastically differ from the stream. Thus it will still return to its original route. As is the case, it will replan its path to pass the obstacle on the right and keep on moving along the stream.

The concept of streams is able to cope with the crossing of different streams. In Figure 19a an example is presented. As the crossing streams are separated by a large angle, agents will hardly be affected by the crossing of others. Once more the blue agent A will mainly be influenced by the red agents. Thus it will keep on following its current stream. Note that some interaction effects do occur. This will cause the agents to cross at a slight angle. This behaviour is exactly as we would expect when two crowds of agents cross.



(a) Crossing streams at a large angle.



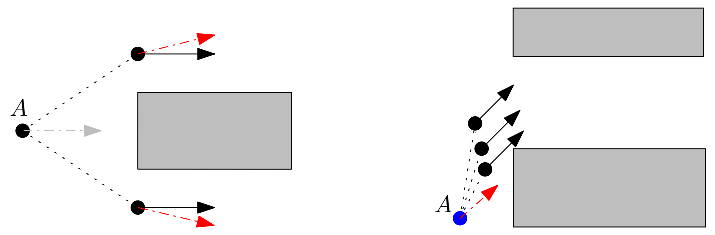
(b) Merging streams.

Figure 19: The effect of following streams in different situations. The agent A in question is drawn in blue. Important neighbours are represented in red. Streams are represented by dash-dotted lines.

When two streams move in approximately the same direction, they will merge and form one larger stream. If the difference in angle is small enough, an agent will be affected by all agents in front. This will cause both streams to merge. This can be seen in Figure 19b.

There are still some cases where the detection of streams is troublesome. In Figure 20a an example is given where the resulting stream is clearly not realistic. If two or more agents move on either side of an obstacle, perceived velocities (represented in red) might cancel each other out. This results in a perceived local stream for agent A that will collide with the obstacle.

Even though the local stream is incorrect, the merging technique as discussed in Chapter 5 will make sure a decision is taken. The agent will be more inclined to move along its planned global route. This preference for a side will resolve this problem. Once either decision is taken, the stream approach will once more pick out the correct stream.



(a) Moving to either side of a central obstacle. Black arrows represent actual velocities. Red arrows represent perceived velocities. The dashed, gray arrow at agent A represent the perceived local stream.

(b) Cornering an obstacle by the shortest route.

Figure 20: Troublesome cases can arise when determining the perceived velocity. The resulting stream velocity is clearly unrealistic. Agent A is the agent in question.

Another situation in which stream detection is troublesome occurs when cornering an obstacle. If a group of agents tries to pick a short route around an obstacle, some agents might get into trouble. Once again agent A comes into trouble as the stream will point towards the obstacle instead of towards the hallway. In this case as well, the merging technique will help out. If this is not sufficient, micro-avoidance behaviour (Section 3.3) will prevent agents from colliding with the wall.

5 Merging individual and stream behaviour

In the previous chapter we have defined stream behaviour for agents. However, agents should not always display stream behaviour. If the local density is low agents should display individualistic behaviour. We set out in this thesis to marry both worlds. In this chapter we will define how the transition between both approaches will occur. This transition can be set individually for different agents. By individualizing this meta-strategy, we allow for different agent profiles. This will allow for the ability to make individual decisions which might differ from other agents. A basic example of the importance of this can be found in emergency situations. A police officer or firefighter will be very determined to pick their own route. A panicked victim, however, will blindly follow the mob.

The incentive of an agent will determine to what extent an agent will be determined to follow its preferred route. A low incentive will cause stream behaviour. Agents will have a tendency to follow the stream. A high incentive, on the other hand, will cause individual behaviour. Section 5.1 will describe the notion of incentive. In Section 5.3 we will discuss the tendency of humans to be less attentive of the surroundings when moving along with a stream. This behaviour causes agents to react later to upcoming collisions and causes agents to be less inclined to make drastic changes in their current direction of movement. In Section E.1 we will discuss the importance of locally updating the preferred direction.

5.1 Incentive

All agents in the simulation have their own private goals. Consequently, agents have different motivations to display or ignore group behaviour. Agents follow streams as long as this is in their advantage. As already discussed in Section 4.1, a clear example of this behaviour can be seen at train stations. When merging both individual behaviour and group behaviour, it is important to know when to switch between both strategies. To avoid notable transitions between both strategies, we will propose a graduate change in behaviour. The main decision variable in this will be the incentive of the agent.

Definition 8 (Incentive) *The incentive λ_A ($\lambda_A \in [0, 1]$) of an agent A is a measurement of its desire to move along its preferred route. The incentive value is at its maximum value of* 1 *if the agent if fully commited to move along its own preferred route. The minimum value of* 0 *implies that an agent is fully commited to coordinate its movement with the surrounding crowd. The incentive can be influenced by four factors, namely* deviation*,* local density*,* internal motivation *and* time spent*.*

The incentive of an agent will make it pick the most efficient combination of strategies to move through the environment. It will combine both individual and group strategies to achieve one desired behaviour. The incentive is a measurement of the decisiveness of an agent to follow its personal route. If the agent has a maximum incentive, it is fully committed to its personal route. It is less important for the agent to move along with the crowd. Examples include agents wishing to leave a stream. This might be because the stream is heading in a different direction then they wish to go. If the agent has a minimum incentive, it is fully committed to the stream strategy. It could, for example, locally be so crowded that a personal route is unrealistic.

The incentive determines, for each agent, to what degree it is willing to cooperate with the crowd. Incentive is influenced by four different factors. The combination of these factors will determine to what amount an agent applies both strategies. As the incentive is the main variable joining both individual and group behaviour, we will give an explanation of all four factors. All factors will include an agent-based variable. This variable will allow individual behaviour for this specific factor. Note that each factor only gives 'advise' on the incentive. The most emergent factor of these will be decisive in determining the incentive. Therefore, we will take the maximum of all factors. An exception is the *internal motivation* variable, which will determine the width of the sliding scale for the rest of the variables.

As incentive is an agent-based variable all variables will be related to the same agent. For clarity reasons, we will drop the subscript denoting the agent in the following section. All variables are assumed to be related to the same agent A.

- **Deviation**
  Each agent has a preferred direction α_0 in which it would like to move. This direction is determined by the global path planning algorithm in combination with a local attraction point [28, 5]. The stream-based approach will define a direction α_S in which the local stream is moving. The deviation of the local stream from its preferred route will make an agent decide whether it is worth following the stream. This deviation is measured as the angle between both vectors. If the deviation is low enough, an agent will have a incentive of 0. It is more energy-efficient for an agent to move along with a stream, even though a slight detour is taken.
  There is a threshold value below which an agent will always pick the stream direction. From this threshold upwards an agent will get more determined to follow its own preferred route. Above a maximum angle threshold the agent will always decide to follow its own preferred direction. The maximum threshold is determined by the agent variable b. Note that the closer an agent gets to its goal position, the quicker the deviation will grow. In the end agents will always leave the stream.
  The agent-dependent variable b_{min} will determine the minimum threshold for an agent. The higher this value, the longer an agent will keep following the stream. Only if the deviation becomes extreme, will the agent start leaving the stream. The variable b determines how soon an agent will leave the stream if allowed so by b_{min}. The higher this value, the sooner an agent will depart from the stream.
  The deviation factor f_{dev} is defined as follows[11]:

$$f_{dev} = \begin{cases} 0, & min(|\alpha_0 - \alpha_S|, 2\pi - |\alpha_0 - \alpha_S|) < b_{min} \\ b * \frac{min(|\alpha_0 - \alpha_S|, 2\pi - |\alpha_0 - \alpha_S|}{0.25\pi}, & \text{otherwise,} \end{cases}$$

where

$$\begin{aligned} \alpha_0 &= \text{Preferred direction of the agent;} \\ \alpha_S &= \text{Stream direction at the position of the agent;} \\ b_{min} \in [0, \frac{1}{2}\pi] &= \text{Minimum threshold before deviation influences incentive;} \\ b \in [0,1] &= \text{Agent-dependent variable.} \end{aligned}$$

- **Local density**
  Density[12] has an inverse effect upon an agent's incentive. The higher the local density, the more prevalence group goals get. At a low-density value, an agent will move along its preferred route. No group effects are discernible. At high densities, following the stream is the most efficient. Thus its incentive will be nearly 0. Densities in between will interpolate between both strategies. Note, as density is local, that agents at the sides of a stream can have a higher incentive than those in the middle. This is specifically the case when there exists free space on the boundaries of a stream. Thus agents at the side will leave a stream sooner than those in the middle.The agent-dependent variable c affects how much density

[11] Both directions are described as an angle as opposed to the line of sight $\overrightarrow{H_i}$. Also note that the value of f_{dev} can grow larger than 1. We defer the cut-off till the last step.

[12] A definition of density is supplied in the 'Definitions and Formulas' section in the addenda of this thesis.

influences the strategy choice of the agent. If this value gets larger, the agent will be less inclined to follow streams. The local density will have to be higher before the same choice of strategy is made.
The density factor, f_{dens}, is defined as follows:

$$f_{dens} = \begin{cases} 1, & D = 0 \\ \frac{c}{D+c}, & \text{otherwise,} \end{cases}$$

$$\begin{aligned} &\text{where,} \\ D \in [0,1] &= \text{Local density at the position of the agent,} \\ c \in [0,1] &= \text{Agent-dependent variable} \end{aligned}$$

- **Time spent**
  The more time an agent spends to reach its goal, the more it will become set on following its preferred direction. Hence, the less it will take notice of group desires and the higher its incentive will get. When computing its global path, an agent estimates its travel-time based on expected densities. As far as the agent can perceive these densities the expected densities will match the actual densities. For the rest of the route the regular densities are applied.
  As long as an agent has spent less time travelling, time will not contribute to its incentive. As soon as an agent has spent more time than it expected, its incentive will grow. We have set the maximum threshold at two times the expected travel time. From this moment onwards its incentive will be at a maximum.
  The agent-dependent variable d controls how much more time an agent is willing to spend as compared to its estimation. At the lower limit of 0, time is not an issue for the agent. The time-factor will not contribute to its incentive. At the upper limit of 1 an agent will be fully committed to its preferred route once one hundred percent more time is spent than the estimated time. For all settings, time will only become an issue once more time is spent than expected beforehand.[13]
  A possible extension to this factor is to change its expected travel time when replanning. This is, however, a speculative extension and not required for the global method. When an agent replans its route, it can change its estimation of how long it will take. We assume that agents are reluctant to change their estimation. The new estimated finish-time is the average of the old estimation and the newly calculated estimation. Suppose a route is re-planned and now takes longer than expected before. The agent would be slightly irritated already and will be more determined to follow its preferred route. This is consistent with the effect of estimating the finish time slightly too optimistic. On the other hand, if a route takes less time than expected, an agent will display a more relaxed attitude. This relates to estimating the finish time slightly too pessimistic.
  The time factor f_t is defined as follows:

[13]We note that raising the agent's incentive after the expected amount of time has passed might be too late. A possible alternative is to compare current expected total trip length (actual time spent plus the expected required travel time still needed) with the expected travel time beforehand. This will make an agents incentive rise as soon as a delay is encountered.

$$
\begin{aligned}
f_t &= \begin{cases} 0, & t_{cur} < t_{estFinish} \\ d * \left(\frac{t_{cur} - t_{lastPlan}}{t_{estFinish} - t_{lastPlan}} - 1 \right), & \text{otherwise,} \end{cases} \\
&\text{where} \\
t_{cur} &= \text{Current simulation time;} \\
t_{estFinish} &= \text{Estimated finish-time for the current route;} \\
t_{lastPlan} &= \text{Time of last global route-planning;} \\
d \in [0,1] &= \text{Agent-dependent variable.}
\end{aligned}
$$

- **Internal motivation**
  An agent will have a basic internal motivation to comply to the social pressures of a crowd. A hurried agent or a rescue services personnel member might have a lower motivation to comply. Their incentive will consistently be higher, no matter what the circumstances are. The internal motivation of an agent determines a minimal incentive value that an agent at all times has. Circumstances may cause it to increase, but it will never decrease. Internal motivation is simply defined as a.

All four factors contribute to the incentive of an agent. We assume that the most dominant one will have the most effect on an agent's behaviour. Therefore, to compute the incentive, we will take the largest of the first three factors. We will limit the resulting value at a maximum of one. The fourth factor (internal motivation) will be used as the minimum incentive possible. As the maximum value is bound at 1, the other factors are scaled by $(1 - a)$. To compute the incentive λ_A of an agent A, we are left with the following formula:

$$\lambda_A = a + (1 - a) * min(max(f_{dev}, f_{dens}, f_t), 1) \tag{3}$$

5.2 When to coordinate movement

Depending on the density, agents should coordinate up to a higher or lower degree. However, streams should only form if there is sufficient reason. A single group of agents moving in the same direction need not form streams. The contraction that implicitly occurs in streams is counterproductive. Agents should make full use of the available space instead of contracting to coordinate movement. Therefore, we will make an addition to our approach as described above. Agents should only start to form streams if they believe there is sufficient reason. As long as all agents roughly move along in the same direction no need is present. However, if an agent is detected that is moving in a different direction streams start to become important. The direction of the agent should be at least perpendicular to the agent's direction of movement for it to move in a significantly different direction. If no such agent can be detected in the field of view, than no streams need to form. Thus, the incentive of the agent is set at 1, causing it to follow its own route.[14]

We note that this choice has a partially undesired effect on the formation of streams. Multiple streams running in nearly the same direction can now no longer affect each other. This does, however, generate interesting new emergent behaviour. We will discuss this in more detail in the future work section (Chapter 8).

[14]In our implementation, we have chosen to check only the set of nearest neighbours. This set of neighbours might be too restrictive in dense environments. We will only be able to perceive the nearest x agents. Therefore, we also react if one of these agents perceives an opposing agent.

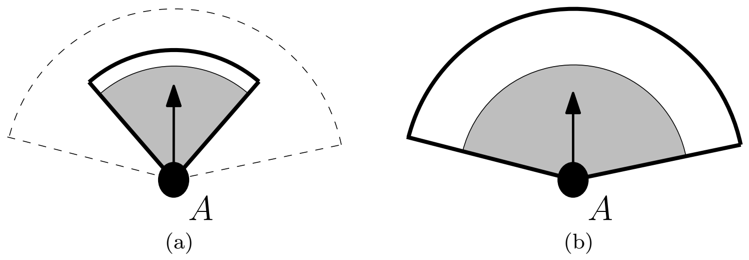


Figure 21: Attentiveness of an agent. The total field of view is represented by a thick, black line. The set of allowed velocities is displayed in grey. The dashed area is the maximum viewing range. (a) Field of view for an agent with low attentiveness (b) Field of view for the same agent with full attentiveness.

5.3 Attentiveness

When agents move along in a stream, they become less attentive of their surroundings. This behaviour is visible in two tendencies that agents display. The first is a tendency to react later to an upcoming collision threat. Once the threat is recognized, agents have a higher aversion to deviating from their current course. Both tendencies relate to the agent's field of view (FoV). An agent will look less far ahead and will consider a smaller cone of the environment. The most extreme result of this effect can be detected in high-density environments, where persons only look at the feet of the person ahead. We will define the combination of both tendencies as *attentiveness*.

Definition 9 (Attentiveness) *An agent's attentiveness is defined as its consideration of its environment. A fully attentive agent will keep track of upcoming collisions with all objects in its maximum field of view. The full range of allowed velocities is available to avoid these collisions. As agents become less attentive, their field of view will decrease and the set of allowed velocities is also decreased. Mathematically, attentiveness is defined as*[15]*:*

$$attentiveness = max\left(min\left(\gamma * \frac{D_o}{D_a}, 1\right), \lambda * \delta\right),$$

$$\begin{aligned}
&\textit{where}\\
D_a \in [0,1] &= \textit{Local density, when measuring only agents with an } \text{alongside velocity;}\\
D_o \in [0,1] &= \textit{Local density, when measuring all other nearby agents;}\\
\lambda \in [0,1] &= \textit{Incentive of the agent;}\\
\gamma \in [0,1] &= \textit{Agent-dependent variable;}\\
\delta \in [0,1] &= \textit{Agent-dependent variable.}
\end{aligned}$$

Agents are supposed to have an alongside velocity *if* $\hat{\mathbf{v_A}} \cdot \hat{\mathbf{v_B}} \geq 0.5$.

The attentiveness of an agent is defined as a value between 0 and 1. The attentiveness directly determines what percentage of its maximum viewing range it is using. It also determines the percentage of the maximum viewing angle that is used. The agent can still move up to its maximum speed, but the allowed velocity directions are bounded by its maximum viewing angle. A visual depiction is given in Figure 21. This setup effectively causes the agent to display both behaviours. Agents will be less attentive of their surroundings and will take less extreme

[15]Note that this implies that attentiveness can become 0. In practice a minimal attentiveness needs to be ensured.

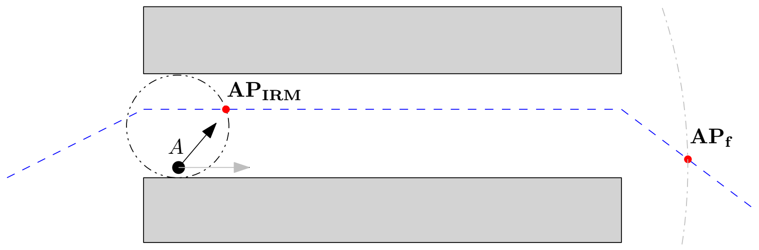


Figure 22: In narrow corridors, the IRM method can deliver undesired incentive values. This is because the attraction point is not located in the expected direction. Looking ahead can alleviate this problem. The indicative route of an agent A is displayed in blue. The original attraction point $\mathbf{AP_{IRM}}$ and the new attraction point $\mathbf{AP_f}$ are displayed in red.

actions to avoid collisions. It will also force the agent to be more inclined to keep moving along the stream when in a high-density stream. This is a desirable effect, as a high-density stream implies a high-density surrounding (see Section 4.2.3). We do not want agents to keep trying to move out of the stream, but we want a high level of coordination.

When moving upstream of a large crowd, an agent has full collision avoidance capabilities. The value of $\frac{D_o}{D_a}$ will be large (or infinite), causing an attentiveness of 1. These avoidance capabilities are needed because all opposing agents will be less inclined to provide collision avoidance as $\frac{D_o}{D_a}$ is very small. This behaviour fits with the displayed tendencies of crowds.

As can be seen from the definition, an agent's attentiveness is dependent on two factors. It should currently be following a stream and it should be inclined to follow this stream. Only if both are the case, do we want the agent to become less attentive. If an agent decides to leave a stream, its attentiveness should grow once again. This will allow it to safely leave a stream.

A nice side-effect of this approach is that computational costs drop when coordination rises. This is expected to happen in crowded areas where a lot of computations are normally required.

The agent-dependent variables γ and δ determine the relative importance of both these factors for an agent. A relatively low value of δ will cause an agent to become less attentive as soon as it perceives it is moving along a stream. This is independent of whether it is actually following this stream. So an agent will remain less attentive when exiting a stream. A low value of γ will cause an agent to only lower its attentiveness if it is purposely moving along in a stream. From our preliminary experiments it turned out that a setting of $\gamma = 1$ and $\delta = 2$ gave the best results.

5.4 Combining Streams with the Indicative Route Method

The combination of incentive and attentiveness causes an agent to move efficiently in both high and low densities. However, an implicit assumption is hidden in the way the incentive is determined. To determine the incentive a comparison is made between the preferred direction and the stream direction. It is implicitly assumed that an agent always knows its preferred direction. The local preferred direction, however, might not be in line with the globally preferred direction. An example where this is not the case is in narrow corridors. In these cases the attraction point may not be aligned with the expected direction.

In Figure 22 an example is given. Due to the width of the corridor, only a small clearance disc is allowed. This causes the attraction point to be located at a relatively small distance from the agent A. Agent A is pulled towards the top of the corridor. Normally this is not a problem as the agent should return to its indicative route. But in high densities this might cause problems.

When a large group of agents traverses the corridor, a stream forms. Let us suppose that a group of agents moves from left to right. The direction of movement is irrelevant and has been

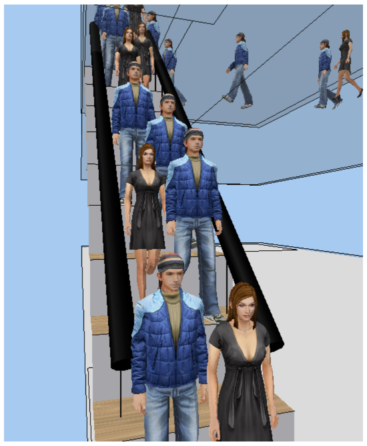


Figure 23: The zipper effect on a staircase. There is not enough space to stand side by side. This causes agents to form up in a zipper like formation. The effective use of the staircase is thus maximized.

chosen arbitrarily. In this case it is reasonable to assume that the stream is roughly directed to the right. It is natural to assume that agent A will move along with the stream as its intuitive attraction point is $\mathbf{AP_f}$. However, its current attraction point $\mathbf{AP_{IRM}}$ is located in a different direction. Locally, agent A will not want to move in the direction of the stream. As there is a large deviation between preferred direction and stream direction, its incentive will be high. So instead of following the stream, the agent will push its way through the crowd. When all agents move in the same direction this is not a major concern. The same scenario could be constructed, however, with two opposing streams of agents. If an agent leaves the stream, it will end up blocking the opposing stream. This is exactly what we try to prevent by using streams.

Two options can help agents pick a more natural attraction point. In the next section we discuss altering the attraction point to fit the general direction of a straight corridor. This alteration is also part of our setup for the experiments. In Section E.1 a different approach is discussed selecting the furthest visible attraction point for each agent. We did not use this option in our experiments.

5.4.1 Moving along the corridor

In some sections of the corridor the exact position of the Indicative Route (IR) is irrelevant for our purposes. A section of the corridor that is bounded by two walls, defines an implicit walking direction. There is no need to move towards the IR. The corridor itself already defines the direction of movement.

In these cases we will alter the agent's attraction point. The agent should keep its relative position in the corridor. Thus, we will pull it along the general direction of the corridor. By altering the agent's attraction point in this way, we prevent unrealistic cross-movements across narrow corridors even further. As a result of this altered attraction point, new emergent behaviour start to occur.

On staircases, elevators and in small hallways a zipper effect occurs (see Figure 23). This behaviour is also perceived in real-life as also shown by Daamen [62].

6 Experiments

We have conducted several experiments to verify our proposed approach. We will start with a set of experiments dedicated to finding the best setup values for our approach. In the second part we will compare our approach to the current state-of-the-art algorithms. The last experiments will be dedicated to showing the abilities and limitations of our approach.

Description of setup All experiments have been run on a single PC. It contained a Intel Core(TM) 2 Due CPU E8400 (2 CPUs at 3.00 GHz). The machine had 3 GB of RAM memory. The graphics card was a GeForce 7600GS with 256 MB onboard DDR-2 memory. The operating system was Windows XP Professional 32-bit with Service Pack 3 and DirectX 9.0c.

All our experiments were run on a single core and without visualisation.

Description of experiments

For our experiments we will stick to a number of scenarios to test the results. We will give a short description of the experiments in this section. A visual depiction of all experiments can be found in the Appendix. All scenarios with the label *Steerbench* are taken from the paper on Steerbench [61].

Circle
A group of agents (32) situated on a circle all try to reach to opposite side of the circle.

Crossing-streams
Two dense groups of 100 agents (streams) cross at a perpendicular angle. The goal of this scenario is to test whether different streams are capable of crossing without heavy interference.

Hallway-one-way
A distributed group of 200 agents try to move along a corridor in the same direction. (*Steerbench*)

Hallway-two-way
Two distributed groups of 100 agents each try to reach opposing side of the same corridor. (*Steerbench*)

Merging-streams
A dense group of agents (stream) is joined by a second dense group of agents moving in nearly the same direction. We are interested to see if both streams merge. (A total of 225 agents are present)

Narrow-hallway-continuous
A narrow corridor of $4m$ wide is populated by two distributed groups of 25 agents moving to opposite sides of the corridor. The scenario is closely related to the hallway-two-way scenario, but sports a higher density.

Narrow-hallway-x
Dense groups of x agents (10, 50, 100) enter a narrow corridor of $4m$ wide trying to reach opposite ends of the corridor.

Oncoming-groups
Two groups of 6 agents move in opposite directions. (*Steerbench*)

For all our experiments we have executed 50 runs for each settings. We have measured the standard deviation to ensure that the results were significantly valid. All distributions over the runs correspond to a normal distribution, as was shown by a QQ-plot of the data. As the data complied to a normal distribution, we were able to run t-tests to determine if the data were significantly different.

Steerbench

To benchmark our steering behaviour we used the benchmark Steerbench [61] as part of Steersuite v1.3 [63]. We have updated the Steerbench framework in some notable areas. For completeness we will state the most important of these,

1. Agents that are set as disabled, are removed from the space grid. This prevents unnecessary overflowing of arrays;
2. Goal positions can be added that lie randomly within a given area;
3. In the original benchmarking score a measurement of kinetic energy was used that was dependent on the frequency of agent updates. If agent information was saved more frequently, scores would increase drastically. As this is clearly unrealistic, we have changed to a different kinetic energy score. The correct kinetic energy was already computed by Steerbench, but not yet used.

We have used the main benchmarking score as provided by Steerbench. This score is comprised of three different components. The average number of collisions c per agent is penalized. The average kinetic energy e_k spent as well as the average time t spent by an agent is also counted against. The total benchmarking score is computed as

$$score = 50 * c + 1 * e_k + 1 * t \,.$$

The resulting score should be minimized by the algorithm tested. That is, for all benchmarking scores a lower score is better. Note that there is an absolute minimum score that is required to get the agents to their respective goals. *We can calculate this minimum score and we will use this as our base mark for any graphs presented in the next part.*

6.1 Determining base variables

For the first part of our experiments we will try to define appropriate settings of all variables. The first experiment will define the optimal combination of base components for the stream concept. We will compare different density measurements and two different micro-algorithm approaches. The next two experiments are aimed at the variables defining perceived velocity and incentive. The base settings obtained through these experiments will be used in all the following experiments.

6.1.1 Density Measurement and Micro-avoidance algorithm

The streams algorithm is dependent on its base components to function properly. Most important amongst these are the micro-avoidance algorithm and the density measurement. To correctly compare the streams algorithm with plain micro behaviour we need the proper base components. In this first experiment we compared three different micro algorithms and different density measurements. From this experiment we hope to conclude which combination functions best. This setup will be used to conduct any further experiments. We will first describe the different core components. We will then compare the different approaches.

Components

We compared three different micro-avoidance algorithms, namely the algorithms by Karamouzas et al. [64], the algorithm by Moussaïd et al. [20] and RVO by van de Berg et al [3]. The *algorithm by Moussaïd et al.* is extensively described in Section 3.3. The reader is referred to this section for further information.

The *algorithm by Karamouzas et al.* is a vision-based algorithm. A field of view is defined in the direction that an agent A is moving. This field of view has a maximum angle of 200º. For all the agents residing within this field of view, we compute the time-to-collision with agent A based on their current velocities. This determines how imminent a collision is for agent A. Based on this time to collision a maximum allowed deviation and speed alteration are defined. These deviation values are extracted from real-life experiments. We refer the reader to the original paper for an exact description of these values [64]. The allowable set of velocities is now concisely defined. For all these velocities the first time to collision is computed based on the visible agents and walls. The best combination of angle and speed is selected based on a number of variables. A consideration is made regarding the deviation from the preferred velocity, the required energy and the risk of colliding. If an agent is already colliding with another agent, the energy term is dropped from the formula. The agent should resolve the collision as fast as possible.

The *RVO algorithm* is based on the concept of velocity obstacles. For an agent A a velocity obstacle is created for each neighbouring agent and obstacle. These velocity obstacles define a set of velocities that will cause a collision somewhere in the future. By selecting a velocity that is part of no velocity obstacle, collision-free motion can be guaranteed. In the RVO2 library [65], the ORCA algorithm is applied. ORCA is an extension to RVO where agents split the ownership of trying to avoid a collision. The RVO algorithm is widely applied in many programs. For our experiments we have made use of the RVO library for C++ v2.0.1 with a horizon of 1.

To measure the local density we have tested three different approaches. These approaches are described in more detail in Section C. The first approach is computationally the cheapest. We measure the overall density on an edge of the Explicit Corridor Map (*edge*). We expect this approach, however, to lack local information. The most expressive measurement is the vision-based density measurement (*vision*), however, this is also the computationally most expensive.

The third approach is a compromise between the locality of vision-based density and the computational costs of edge-based density. We measure local density through the use of a density map. Three different grid sizes are tested of $1m^2$, $4m^2$, $9m^2$ respectively (*grid1*, *grid2*, *grid3*).

Scenarios

To test the proper functionality we will need to test different conditions. At normal conditions, the basic micro-avoidance behaviour should still function properly (*hallway-one-way*). Basic agent avoidance and an effective use of the available space should be the case. We also require the lane formation behaviour at normal densities (*hallway-two-way*). On the other hand, we need to test whether streams provide correct behaviour at high densities. Coordination due to streams should improve agent movement and thus contribute to realistic behaviour (*crossing-streams*, *merging-streams*, *narrow-hallway-50*). This behaviour is even more important as current micro-avoidance algorithms are often incapable of solving these high-density situations.

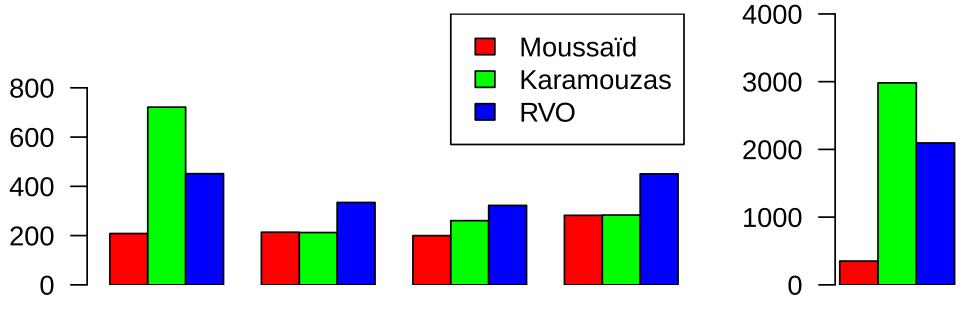
Results

We first compare our implementation of the algorithm by Karamouzas and of the algorithm by Moussaïd. From the results (see Figure 24) it becomes clear that the algorithm by Karamouzas equals the algorithm by Moussaïd in low to medium density situations. In both the hallway-one-way as the hallway-two-way scenarios both algorithms perform equal. In the merging-streams scenario, which sports uni-directional traffic, scores are equal as well. However, in high density or troublesome scenarios, such as the crossing-streams or narrow-hallway-50 scenario, the algorithm by Moussaïd surpasses the algorithm by Karamouzas. In our experiments we noticed that the algorithm by Karamouzas has trouble in narrow hallways. The algorithm can not correctly discern walls from characters if the time to collision is small for both. As we are especially interested in high-density cases, this is troublesome for the goals we set. A second argument that favours our selection of the algorithm by Moussaïd our the running times (see Table 1). The algorithm by Moussaïd has a near constant runtime that is independent of the local density. The algorithm by Karamouzas is far more vulnerable to densities. As density increases (crossing-streams and narrow-hallway scenario) the required computations increase heavily, thus slowing the algorithm down.

The RVO algorithm is less informed than the other two algorithms. The algorithm is, however, very quick and shows a nearly constant runtime. We notice that this algorithm has trouble in highly dense scenarios. Agents are not necessarily inclined to keep any distance to other agents. This can quickly cause trouble in higher densities. Both the algorithm by Karamouzas as RVO are in all runs unable to solve the narrow-hallway-50 scenario. It would be interesting for future work to compare RVO, when a given distance is enforced. Nevertheless, we maintain our choice for the algorithm by Moussaïd et al. The ability to inherently have a better handle on higher density scenarios dominates the slightly higher runtimes.

	Moussaïd	Karamouzas	RVO
Crossing-streams	3.4	11.8	1.0
Hallway-one-way	2.7	2.4	1.3
Hallway-two-way	2.7	5.7	1.2
Merging-streams	2.1	9.9	2.1
Narrow-hallway-50	2.7	17	1.2

Table 1: Average time (ms) required for one step of the simulation algorithm as extended with the streams algorithm. A single step of the simulation algorithm updates all agents. Times were averaged over the total number of simulation steps in a run. The results from 50 runs were averaged per scenario.



From left to right: crossing-streams, hallway-one-way, hallway-two-way, merging-streams, narrow-hallway-50.

Figure 24: Comparison of the mean scores over all density measurements for the different algorithms. The density measures tested are the edge-based density, grid-based density and vision-based density. Both the algorithm by Karamouzas and RVO were not able to solve the narrow-hallway-50 scenario.

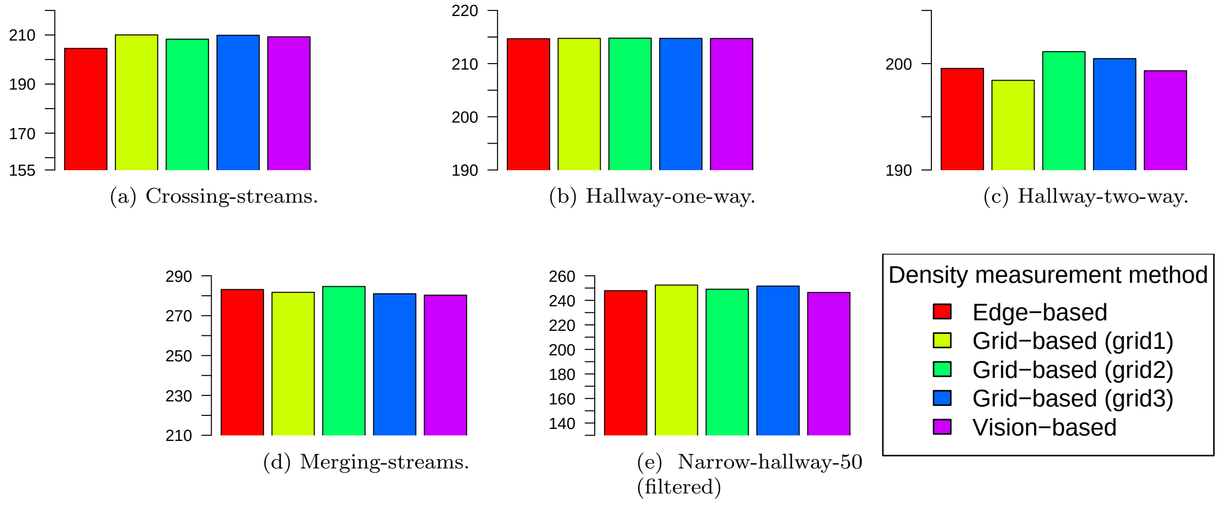


Figure 25: The average benchmarking scores over 50 runs for the different scenarios and density settings. The micro algorithm by Moussaïd is used. Edge-density is represented in red, grid1 in yellow, grid2 in green, grid3 in blue and vision in purple. The scores for the narrow-hallway-50 scenario have been filtered. All runs taking longer than 85s have been removed from the results before computing the mean score.

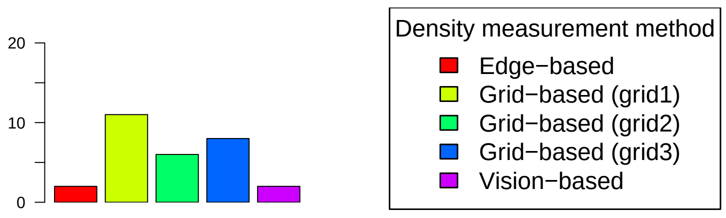


Figure 26: Total number of runs resulting in a deadlock out of 50 runs for different density settings. (Narrow-hallway-50 scenario)

As for the density measurements, a less clear decision must be made (Figure 25). When we compare the different density approaches in the 'normal' scenario (hallway-one-way), no difference is discernible (see Figure 25b). This is what we would expect from this scenario. In normal situations, streams do not disturb the regular micro-avoidance behaviour. As streams have no effect, the choice of density measurement is irrelevant. We note that this implies that any change in scores in the other experiments will be directly related to the streams algorithm. There is also no effect discernible when two streams merge (see Figure 25d). This can statistically be confirmed. For all the density measurements the null hypothesis is accepted with a p-value of 5%. In both the hallway-one-way as the merging-streams scenario there are no agents that move in a different direction. Streams are not considered an advantage and are, therefore, not formed.

If we look at the experiments where groups of agents oppose each other (crossing-streams, hallway-two-way), then a change would be expected. In the relatively low-density scenario (hallway-two-way) the grid1-based and vision-based measurements give the best results (Figure 25c). Both distributions are statistically better than the other three density-measurements (for a p-value of 0.5%). We do note, however, that the improvement is only small.

Unexpectedly, in the high-density scenario (crossing-streams) no effect of the different density measurements is noticeable (Figure 25a). For all the score distributions of the density measurements the null-hypothesis is accepted when using a p-value of 5%.

In even higher densities, however, the need for streams becomes clear. The narrow hallway scenario (narrow-hallway-50) is that crowded that it often results in a deadlock (Figure 26). Both the vision-based density as the edge-based density are, however, nearly always able to solve it. We notice the requirement for local density measurement here. Vision-based measurement has a clear advantage over all grid-based measurements. Interestingly enough, the edge-based density measurement also seems to solve the situation. Even though this density measure is effective, it is questionable whether this is due to the situation or the measurement. It is quite easy to think of situations where the edge-based density measurement is unable to solve the situation. We might, for example, think of a piece-wise curved hallway consisting of a set of small edges. Density on these edges is unlikely to represent the actual density ahead. This is, in turn, likely to cause a break-down in the formation of streams.

It should be noted that the scores for the narrow-hallway scenario (as seen in Figure 25e) are not representative. To obtain these scores we have filtered out all scenarios resulting in a deadlock. As deadlocks are the critical factor in this scenario, they are more important than the actual benchmarking scores. Scores might be heightened as deadlocks need to be prevented. This should not be punished.

To filter out all the correct runs we had to define when a scenario was deadlocked. To do so, we have determined a maximum time (85s) within which all agents should be able to reach the goal. If not all agents were able to reach the goal position within this time the run was deemed deadlocked. This does not necessarily imply that the agents were unable to resolve this deadlock. However, the passageway was blocked at some point in time which we deem undesired behaviour.

All in all we conclude that the micro algorithm by Moussaïd compared with a vision-based density measurement is the best setup for our streams approach. We will use this setup for our further experiments unless stated otherwise. Edge-based density appears to be a possible substitute, but as its generalizability is questionable we choose to remain with the vision-based approach.

	β 0	1	2	3	4	5
α *0*	0	4	1	2	2	4
0.2	5	3	7	4	1	2
0.4	4	2	4	4	5	3
0.6	4	4	2	5	5	3
0.8	3	3	2	4	2	2
1.0	3	2	2	3	1	2

Table 2: The number of runs resulting in a 'deadlock' in the narrow-hallway scenario (as related to the settings for α and β out of 50 runs).

6.1.2 Perceived velocity

To determine the local stream we have defined the concept of perceived velocities (Section 4.1). Two variables α and β were introduced to determine the weight of an agent N its perceived velocity (see Equation 2). This weighted perceived velocity is taken into account for the stream as perceived by an agent A. The weight is dependent on the angle between the velocities of both agents. The variables α and β determine the shape of this weighing function. Variable α determines a lower bound on the weight for each agent in sight independent of its direction.[16] The value of β determines how quickly this weight degrades as the angle between the velocities increases. In this experiment we will try to define a basic optimal setting for both variables. For the α variable we will check values between 0 and 1 with increments of 0.2. For the β variable we will test values between 0 and 5 with increments of 1.

Scenarios

We have tested the settings in two scenarios where the correct formation of streams is critical. In the narrow-hallway scenario (*narrow-hallway-50*) the correct formation of streams is critical for the throughput. The available space is so limited that, without coordination, agents end up blocking each other. The *crossing-streams* scenario requires agents to work together to allow passing. However, agents should not start to form a new combined stream.

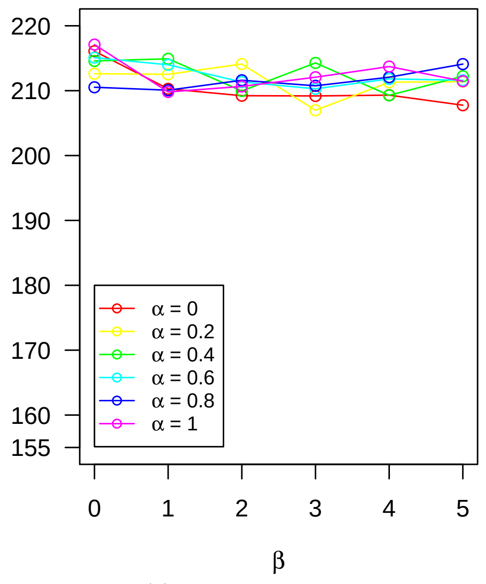
Results

The narrow-hallway scenario is such a troublesome scenario that deadlocks start to occur. These deadlocks cause the results to be skewered. More collisions occur in runs containing deadlocks and travel time is increased largely. On the other hand, kinetic energy is often reduced as agents never reach their destination. To be able to usefully analyse the data, we have filtered out all runs that resulted in deadlocks.[17] The number of runs resulting in deadlocks was also stored.

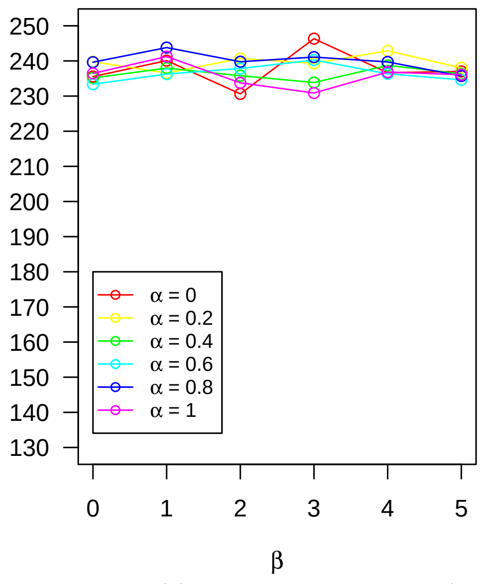
Note, if a scenario is 'deadlocked', this does not necessarily mean that the agents were unable to resolve the situation. In some cases, the 'deadlock' was still resolved within 200s. However, results in these cases were already that skewered that they overly influenced the averages. We will compare the results from the filtered set of runs. As deadlocks are critical in this scenario, we will also compare the number of runs resulting in deadlocks.

Results on the number of deadlocks in the narrow-hallway scenario are displayed in Table 2. A visual depiction of the mean scores for both scenarios are supplied in Figure 27. From our experiments we conclude that the weight of the perceived velocity has almost no impact on the algorithm. It appears that this weighting step can be omitted. In both scenarios the results for

[16]Please note that we never take into account agents that move at an angle larger than 90° as opposed to the subject agents own velocity.
[17]The definition for a deadlocked scenario is supplied in the first experiment.



(a) Crossing-streams scenario.



(b) Narrow-hallway scenario (filtered)

Figure 27: Mean benchmark scores for the different settings of α and β. A lower benchmarking score represents a better result.
Scores for the narrow-hallway scenario have been averaged over all runs not resulting in a deadlock. Runs resulting in a deadlock were filtered out. A scenario is deemed deadlocked if not all agents are able to reach their goal within 85s.

all the possible different settings do not significantly differ (for a p-value of 0.5%). The number of deadlocks does differ depending on the settings of α and β however (see Table 2). The value of $\alpha = 1.0$ gives the most stable results over all values of β. This value coincides with a weight of 1 for all perceived velocities independent of their direction. We conclude from this that it is important to take into account all agents moving in the same general direction. Streams moving in a different direction than the agent should quickly be detected. The value of β has become obsolete by selecting $\alpha = 1$. We will assume $\beta = 0$, but this is not of importance any more.

It is surprising that the weighting values do not influence results. We believe this is due to the fact that we have already filtered out all agents moving in an opposing direction. We have also turned off stream behaviour in uni-directional scenarios. In future research we might want to form streams in these scenarios as well. It is likely that the weighting factors will play a more distinct role in these cases.

6.1.3 Incentive

The incentive of an agent determines to what degree an agent is willing to cooperate with the perceived local stream. It is agent-based and there are a set of agent-dependent parameters that influence the behaviour. These parameters can be set per agent to achieve different behaviours. However, we require a base setting that results in realistic behaviour. Preferably these settings should be able to cope with a wide range of scenarios. We will give a short description of the parameters. For a more extensive description please refer to Section 5.1.

Components

***IncentiveA* parameter**
This parameter determines the minimum amount of incentive an agent has. This might for example be set to model rescue workers, who should not be affected by the collective escape movement of a crowd.

***IncentiveB* parameter**
The effect that a deviation of the stream direction from the preferred direction has. The higher this value, the less deviation is required before an agent start to follow its own course.

***IncentiveBMin* parameter**
A minimum required deviation that is required before an agent will react.

***IncentiveC* parameter**
The effect that density has on an agent can be altered by this parameter. A higher density will result in a lower incentive. A higher value for this parameter will cause the agent to be less affected by densities. Thus the incentive of the agent will remain higher for a longer interval.

***IncentiveD* parameter**
This parameter represents the effect that the 'time spent' has on an agent's incentive. If an agent spends more time than expected to get to the goal, its incentive will start to rise. If this parameter is higher, time becomes more important for an agent.

Scenarios

To test the effect of the incentive values we have selected a wide range of scenarios. The main scenarios that we will be interested in are the high-density scenarios (*crossing-streams*, *narrow-hallway-50*). These scenarios have opposing requirements. In the narrow-hallway scenario, agents should highly cooperate to prevent deadlocks. Overly coordinating motion in the crossing-streams scenario, however, might be detrimental.

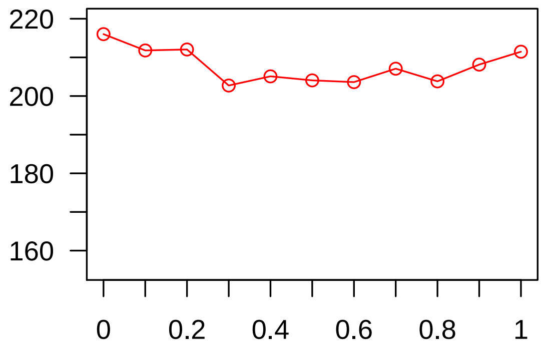
As we want realistic behaviour at lower densities as well, we will look at the *hallway-two-way* scenario. For completeness we also ran the *merging-streams* scenario. No effect should be noticeable here as streams do not occur. As this was the case we will not discuss this scenario any further.

Results

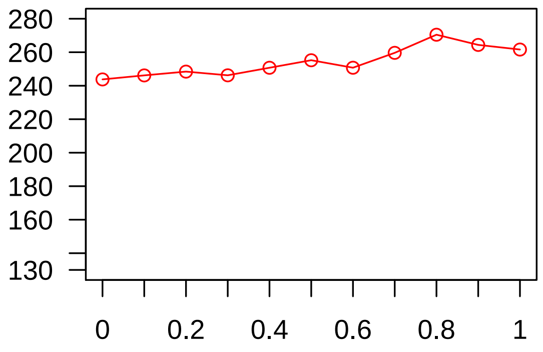
For this experiment we will once more look at the benchmarking scores combined with deadlocking results. The definition of a deadlocked scenario has been provided in the previous experiments. We will discuss the results separately per parameter for clarity. The combination of all parameters will be tested and discussed in the conclusion of this section.

The *incentiveA* parameter is a very influential parameter. We note that it has a different effect on each scenario. In the hallway-two-way scenario no effect is discernible as a consequence of the setting of this parameter. The density in this scenario is not yet of such a level, that more egocentric behaviour is punished. The narrow-hallway-50 scenario, on the other hand, shows a clear decrease in performance as the incentiveA parameter is set at higher values (see Figure 28b). Coordination is essential in this scenario and any lack of such is detrimental for the crowd's motion. The number of deadlocking runs in this scenario paints an even clearer picture that coordination is essential (Figure 28d). As the value of the incentiveA parameter rises, the number of deadlocking runs rises linearly as well. This leads us to believe that coordination is essential.

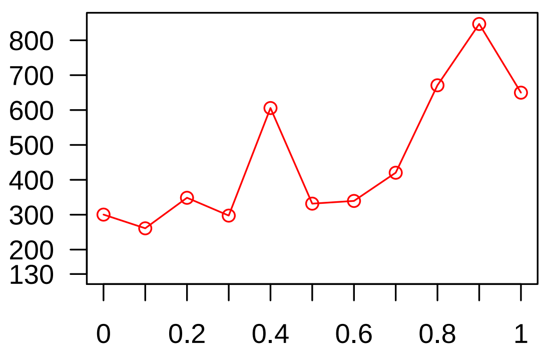
The crossing-streams scenario displays a preference for a slightly higher value of the incentiveA parameter (see Figure 28a). Settings between 0.2 and 0.8 show a significant increase in performance (p-value of 0.1%). In this scenario there appears to be a smaller desire to cooperate movement. The high scores in the crossing-streams scenario at the lower settings are an artefact.



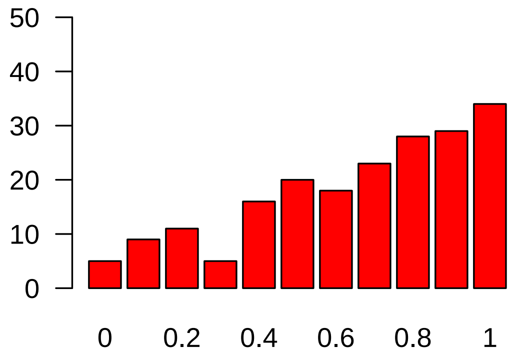
(a) Crossing-streams.



(b) Narrow-hallway-50. (filtered)



(c) Narrow-hallway-50.



(d) Number of deadlocks for narrow-hallway-50.

Figure 28: a-c) Benchmarking scores for as dependent on the incentiveA parameter.
d) Number of deadlocks in the narrow-hallway-50 scenario as dependent on the incentiveA parameter.
Scores for the narrow-hallway scenario have been averaged over all runs not resulting in a deadlock. Runs resulting in a deadlock were first filtered out. A scenario is deemed deadlocked if not all agents are able to reach their goal within 85s.

As agents coordinate more strongly, they start to move in the same direction. However, this causes them to 'lose track of' the agents behind them. These agents have left the field of view and are, therefore, not taken into account any more. Thus, the agent can cross paths once more with this agent, resulting in more collisions. This increase in collisions was backed up by our results. Future work might be done on 'remembering' important neighbours.

It is impossible to select a value that suits both scenarios. As our base value we choose the value of 0. First of all, the narrow-hallway-50 scenario is more likely to occur. More importantly is the extremity of the change in this scenario. The filtered scores of the narrow-hallway-50 scenario show a skewered result. When looking at the unfiltered scores we get clearer picture of the effect of the incentiveA parameter in this scenario (see Figure 28c). This effect is deemed more important than the relatively small increase in the crossing-streams scenario which is likely as well to be caused by an artefact. Lastly, we prefer not to limit the amount of coordination. This is contradictory to the goal we set ourselves. All in all it appears that coordination is essential.

The results for the *incentiveB* parameter are clear-cut. For most experiments run no effect is notable. The setting of incentiveB is irrelevant. Only in the crossing-streams scenario an effect is notable preferring a high setting of the incentiveB parameter (see Figure 29). Therefore, we will select the value of 1.0 for incentiveB. This will intuitively mean that at an angle of 45º incentive will be at the maximum.

For the *incentiveBMin* parameter the crossing-streams scenario is the only scenario showing changes as well. A trend is visible favouring incentiveBMin at 0 (Figure 30). Thus, from our results it appears that no threshold is needed. A threshold value may even harm results in some cases. We will set incentiveBMin at 0 as base setting.

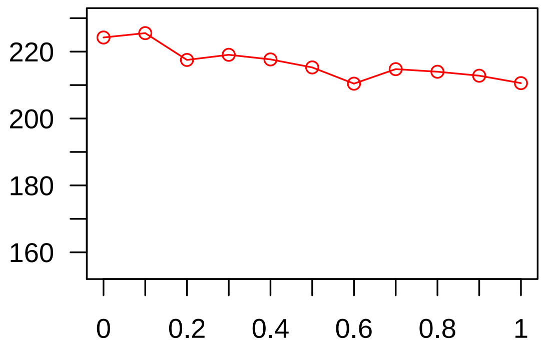


Figure 29: Benchmarking scores as dependent on the incentiveB parameter for the crossing-streams scenario. The lower and higher settings of the incentiveB parameter are significantly different with p-value 0.01%.

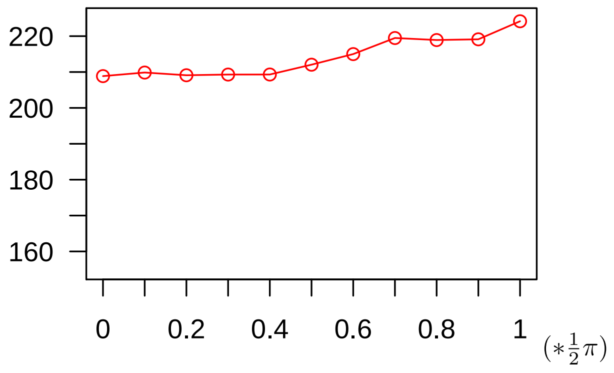
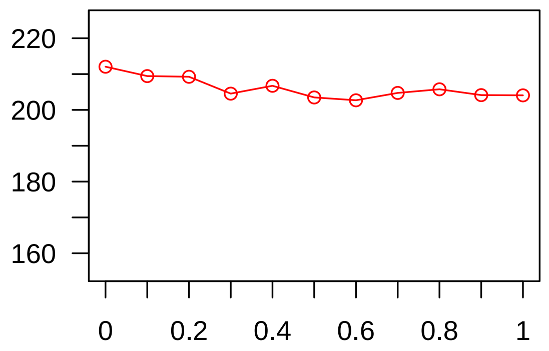
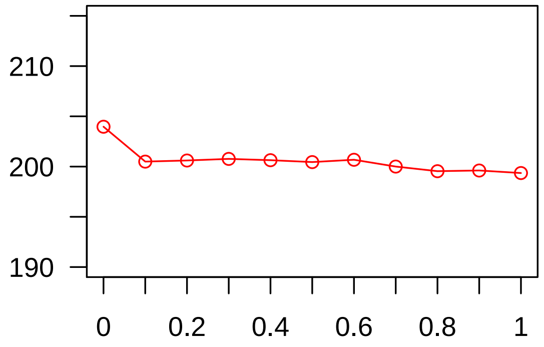


Figure 30: Benchmarking scores for as dependent on the incentiveBMin parameter in the crossing-streams scenario.

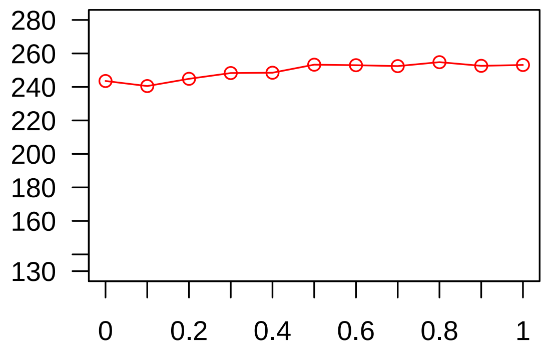
For the *incentiveC* parameter we see a distinction between the different scenarios requiring different settings once more (Figure 31). The hallway-two-way scenario is not affected by the settings of the incentiveC parameter. We perceive a clear preference away from the settings of 0, but other settings show no significant change. In the narrow-hallway-50 scenario an increase in deadlocks is notable as we increase the value of the parameter. A slight increase in score is also perceived in the filtered results for this scenario, but this appears not to be significantly different. The crossing-streams scenario, on the other hand, shows a significant, but small, preference for a high setting of the incentiveC parameter (p-value 1%). This coincides with a far smaller degree of coordination. Once more we note that coordination appears to be counterproductive in this scenario. For our base settings we decide on a low value. Even though this disagrees with the crossing-streams scenario, we believe the increase in the narrow-hallway-scenario to be more important as also discussed before. Therefore, we will select the value of 0.1 for the incentiveC parameter as consistent with the results from the hallway-two-way scenario.



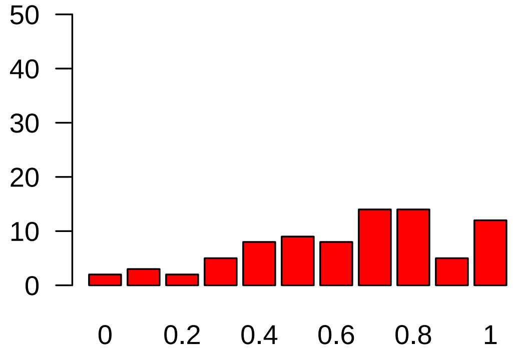
(a) Crossing-streams.



(b) Hallway-two-way.



(c) Narrow-hallway-50 (filtered)



(d) Number of deadlocks for narrow-hallway-50.

Figure 31: a-c) Benchmarking scores as dependent on the incentiveC parameter.
d) Number of deadlocks in the narrow-hallway-50 scenario as dependent on the incentiveC parameter.
Scores for the narrow-hallway scenario have been averaged over all runs not resulting in a deadlock. Runs resulting in a deadlock were filtered out. A scenario is deemed deadlocked if not all agents are able to reach their goal within 85s.

For the *incentiveD* parameter, no result is discernible in our experiments. This is likely due to the size of our experiments. For correct tests larger environments should be tested where agents can experience larger deviations from the expected values. Future research should be conducted on this parameter. For our experiments, we will turn off this setting by selecting the base settings of 0.

6.1.4 Conclusion

From the previous experiments we conclude that the correct settings can make a significant difference on the behaviour of agents. For the next experiments we will make use of the following settings:

- Micro-avoidance algorithm by Moussaïd
- Vision-based density measurement
- Base agent settings:
    - Perceived velocity: $\alpha = 1$, $\beta = 0$.
    - Incentive: $a = 0$, $b = 1.0$, $bMin = 0$, $c = 0.1$, $d = 0$.

To test the combination of these settings we have run all experiments with the optimal settings. The results are displayed in Table 3. These results mirror our expectations based on our previous experiments. We conclude that we can safely combine the results from the different parameters as results do not worsen due to this combination of variables.

	Benchmark score	Deadlocks
Crossing-streams	204	0
Hallway-two-way	198	0
Narrow-hallway-50	241	4

Table 3: Results for the combination of optimal settings (averaged over 50 runs).
Benchmark score is computed with Steerbench. Deadlocks are measured by the time passed before the scenario is resolved.

6.2 Comparing Streams and Micro-avoidance

Having determined the base values for our algorithm, we can now compare it to the basic micro-avoidance algorithms. In this experiment we will run several complicated scenarios. By testing these scenarios we hope to determine when streams form a useful addition to micro-avoidance algorithms. We can not compare between different micro algorithms as they are not equally optimized. Thus, we only compare the different algorithms with their extended versions.

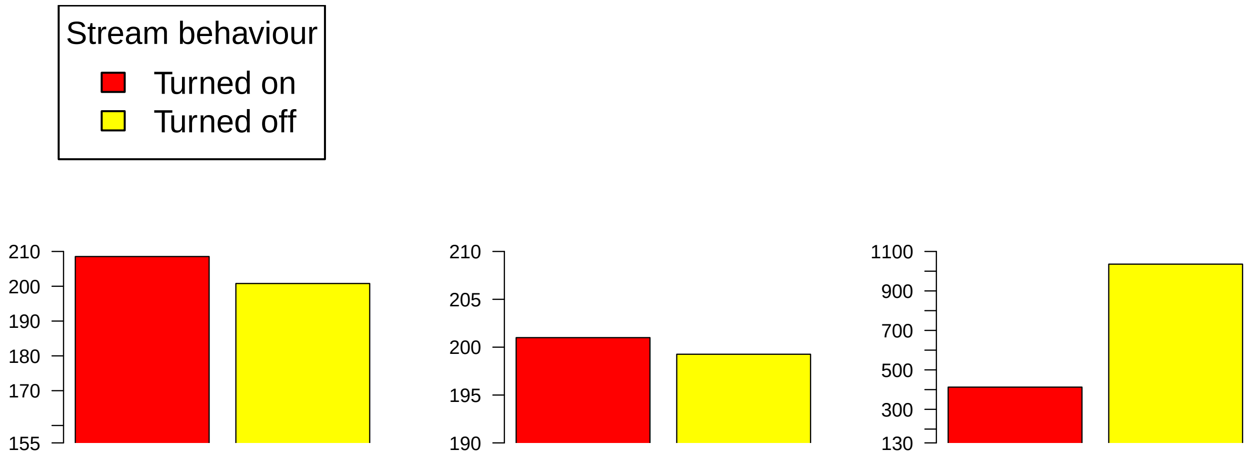
Scenarios

The main scenarios we will run are the *narrow-hallway-50* and the *crossing-streams* scenario. Both scenarios include a very high density and require the correct formation of flows. Besides these two extreme scenarios we will also test the *hallway-two-way* scenario. This scenario is a less dense scenario, showing behaviour in common cases. By testing these scenarios we try to get an overview of the behaviour of the algorithm in different circumstances.

Results

When comparing the micro-behaviour with the streams algorithm two things stand out. First of all, the increase in coordination clearly improves behaviour in the high density scenario (narrow-hallway-50) (Figure 32). For the algorithm by Karamouzas this same trend is visible (Figure 35). However, the stream-extended algorithm by Karamouzas was still unable to solve the situation in many cases.

The second is that the crossing-streams scenario appears to be hurt by the inclusion of streams. For both algorithms we see an increase in score. This result once more appears to point out that coordination does not improve results in this scenario. The higher score is mainly caused by a increase in the number of collisions. We note that several things are the case here.



(b) Crossing-streams scenario with the micro-avoidance algorithm by Moussaïd.

(c) Hallway-two-way Moussaïd.

(d) Narrow-hallway-50 Moussaïd.

Figure 32: Benchmarking scores for different scenarios.
We compare the plain micro algorithm by Moussaïd with the stream extended version.

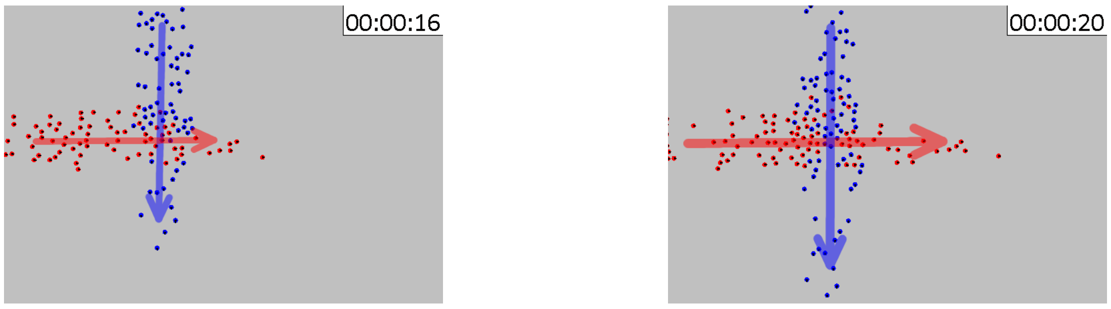


Figure 33: Micro-avoidance algorithm by Moussaïd. Two streams of agents cross. Neither stream has a significant impact on the movement of the opposing stream.

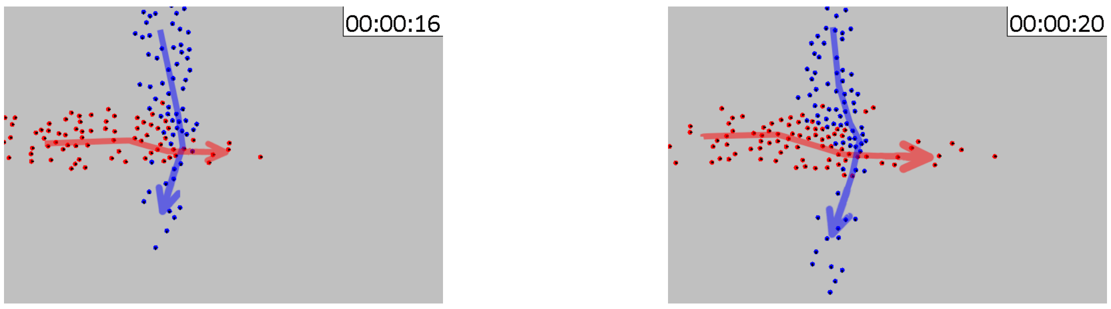


Figure 34: Streams algorithm in combination with the micro algorithm by Moussaïd. Both crossing streams affect the shape of the opposing stream. As a consequence a narrower bottleneck is formed where both groups cross. This causes an increase in collisions.

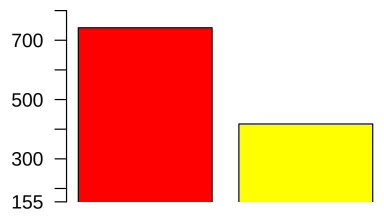
Streams cause agents to alter their direction, to meet in a general direction. From a visual inspection we can clearly see the effect of this (Figure 33 and 34). As agents move along the same lines, they can lose track of each other, as discussed in the previous experiment. This flaw in current vision-based algorithms can cause an increase in collisions.

Nevertheless, even is this was not the case, this behaviour is undesired. Streams cause agents to move less perpendicular through opposing groups. Although this behaviour is desirable, the result is unrealistic. In real life we would expect agents to pick 'holes' against the direction of the flow as well. It would appear that the streams algorithm has trouble coping with this scenario. We do note, however, that this is a very extreme scenario.

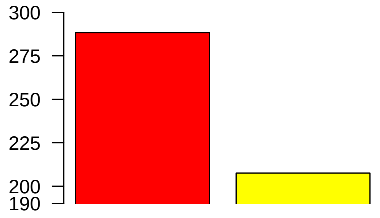
Lastly, we also notice that the streams algorithm equals the performance of the basic micro-algorithm in the hallway-two-way scenario for the algorithm by Moussaïd. Deviation in scores is slightly higher for the stream extended version than for the basic algorithm. This is probably due to a slight increase in collisions as agents start to move more closely together.

The algorithm by Karamouzas heavily favours the plain micro algorithm in the hallway-two-way scenario (Figure 35). A visual inspection makes clear this is due to an increased lane formation. Locally higher densities start to occur and the algorithm is unable to correctly steer all agents if opposing groups of these agents meet.

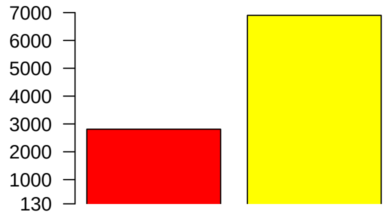
The RVO micro-avoidance algorithm displays the same behaviour as the algorithm by Karamouzas. In the hallway-two-way scenario, results decrease as a consequence of increased local densities. The number of collisions increases largely giving rise to a higher score. The narrow-



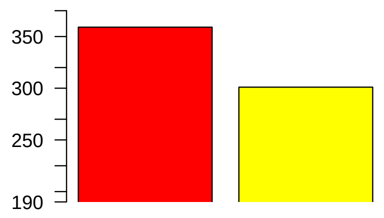
(a) Crossing-streams Karamouzas.



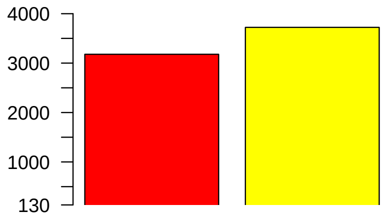
(b) Hallway-two-way Karamouzas.



(c) Narrow-hallway-50 Moussaïd.



(d) Hallway-two-way RVO.



(e) Narrow-hallway-50 RVO.

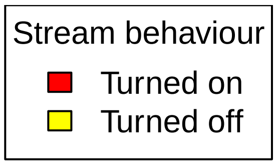


Figure 35: Benchmarking scores for different scenarios.
We compare the plain micro algorithm by Karamouzas and RVO with their respective stream extended version.

hallway scenario displays an increase in performance be it relatively small.

We conclude that streams can improve the behaviour of agents in high-density situations with two-directional traffic. At lower densities the stream algorithm causes an increase in local density due to increased lane formation. Depending on the micro-avoidance algorithm applied this can lead to undesirable results. Some situations still require attention and should be looked at in further research.

To measure the effect on the running time of the streams algorithm we need to take into account the total time spent by the micro-avoidance algorithm. The *attentiveness* of an agent has an effect on the number of neighbours perceived. This in turn can speed up the regular micro-avoidance algorithm. As can be seen in Table 4, the addition of the streams algorithm requires an extra 0.2ms per simulation step on average. For a high simulation rate of 10 frames per second this equals 2ms per second. The extra time required is only a fraction of the required time for the micro-avoidance algorithm. This extra overhead does not appear to strain the running time of the micro-avoidance algorithm.

	Stream behaviour on	Normal algorithm
Crossing-streams	3.3	3.1
Hallway-two-way	2.9	2.7
Narrow-hallway-50	2.8	2.7

Table 4: Average time (ms) required for one step of the simulation algorithm by Moussaïd with and without the streams extension. Times were averaged over the total number of simulation steps in a run. The results from 50 runs were averaged per setting.

The algorithm by Karamouzas displays the same 0.2ms required to run the streams algorithm (Table 5). In the crossing-streams scenario this computation time rises to 2.6ms. This is mainly due to the fact that the streams algorithm forces agents to cross at a smaller choke point. As the agents move closer together, all agents are required to compute collision times for a larger set of speed options. This has a large impact on computation times as already discussed in Section 6.1.1.

The RVO algorithm appears to be unaffected by the addition of the streams algorithm (Table 6). This would imply that the extra time spent by the streams algorithm is returned by the lower computational costs of the main algorithm.

	Stream behaviour on	Normal algorithm
Crossing-streams	12.2	9.6
Hallway-two-way	2.9	2.7
Narrow-hallway-50	2.8	2.7

Table 5: Average time (ms) required for one step of the simulation algorithm by Karamouzas with and without the streams extension. Times were averaged over the total number of simulation steps in a run. The results from 50 runs were averaged per setting.

	Stream behaviour on	Normal algorithm
Crossing-streams	1.0	1.0
Hallway-two-way	1.2	1.2
Narrow-hallway-50	1.2	1.2

Table 6: Average time (ms) required for one step of the RVO algorithm with and without the streams extension.

6.3 Abilities and limitations of Streams

In this experiment we will try to sketch a few scenarios to show the abilities and disabilities of the streams algorithm. We will compare the results of a plain micro-algorithm with the stream-extended version. In contrast to the previous experiment, scenarios have specifically been chosen to be fit or misfit for the stream algorithm. We expect streams to be able to solve all structured situations independent of the density. Unstructured situations, such as criss-cross behaviour, are expected to be destructive for the streams algorithm.

Scenarios

We will test three different scenarios. First we will show the behaviour of the streams-algorithm in different densities. To do so, we test the narrow-hallway scenario under different circumstances. We will run this scenario with oncoming groups of agents of size 10, 50 and 100. These test will show behaviour at low, high and extreme densities respectively (*narrow-hallway-x*). We will also test a less extreme scenario where agents enter continuously. This scenario relates to the hallway-two-way scenario but in a smaller environment.

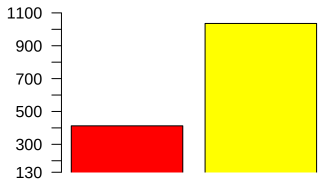
Secondly we will test the *oncoming-groups* scenario as provided by Steerbench. This is a simple and basic test to display that streams do not affect low level micro behaviour.

Lastly, the *circle* scenario is a clear scenario where agents move in all directions. We expect the streams algorithm to be troubled by this scenario.

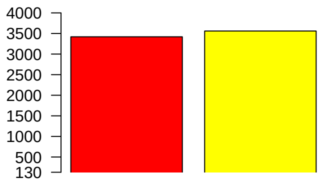
Results

As expected in both low-density scenarios no change in benchmarking performances was noticeable. Both the oncoming groups scenario and the narrow-hallway-10 scenario are unaffected by the streams-algorithm. In the high density density scenario (narrow-hallway-50) we see a clear improvement by the streams algorithm (Figure 35b). Do note, however, that these scores are unfiltered. Thus, deadlocks are the cause of the great increase in score. This is exactly what we need to know, as we need to prevent these deadlocks. In the narrow-hallway-continuous scenario we note a slight improvement in scores (Figure 36d) which is statistically different for a p-value of 1%. As the standard deviation remains the same there is a consistent improvement of score.

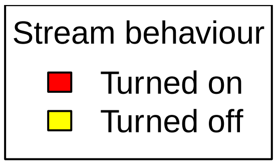
The circle scenario is expected to be the Achilles heel of the streams algorithm. As can be seen from the test-results the streams algorithm is a unable to resolve this scenario. Agents move in a incoherent fashion through the environment. All agents attempt to extract streams from their neighbours, but the resulting stream is clearly incorrect and forces the agent to move further towards the centre. We note that there is in fact insufficient reason to start to cooperate groupwise, making the appliance of streams counterproductive.

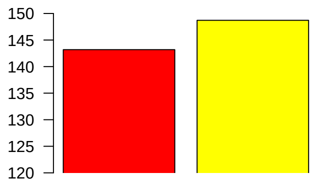


(a) Narrow-hallway-50.

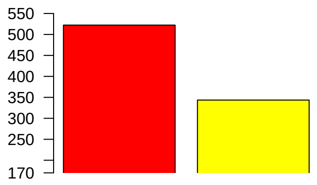


(b) Narrow-hallway-100.





(d) Narrow-hallway-continuous.



(e) Circle.

Figure 36: Benchmarking scores for the algorithm by Moussaïd et al. with and without streams. Benchmarking scores were averaged over 50 runs.

Unexpectedly enough, the streams algorithm has only a minor effect on the extreme density scenario (narrow-hallway-100). This was unexpected as a previous iteration of our implementation of the algorithm by Moussaïd resolved this situation at all times. Due to a lack of time, however, we are unable to further investigate the reasons for this. A visual inspection appears to indicate that it is caused by agents being unable to come to a full stop in time. Collision forces cause them to exit the stream involuntary thereby obstructing the existing other streams causing a deadlock.

We conclude that the streams algorithm is helpful in high-density two-directional situations. Situations that have no structured movement of agents are liable to cause trouble for the streams algorithm. The most problematic scenario encountered so far is the crossing-streams scenario. This is an extreme scenario that we do not expect to come across to often in real life. In fact, most streams of agents crossing in this way would be distributed differently. Due to inherent distribution patterns in arrival and random human behaviour more 'holes' are expected to occur in these streams. These holes help prevent the occurring clog, thus solving the situation.

6.4 Agent Profiles

The flexibility of the agent parameters allows for the generation of different agent profiles. By including different agent profiles in a crowd, we create more life-like crowds. A simple example of this is the inclusion of a leader-follower model in a crowd. The basic settings of the streams algorithm cause all agents to cooperate in their movement. However, in real-life not all agents will cooperate. Agents might decide to leave a stream and follow their own route. These agents in turn cause agents behind them to follow, generating new on-the-fly streams. We have run a basic scenario to test the effect of these different agents. The scenario is based on a real-life crowd experiment run by the Pedestrian Dynamics group at the Jülich Supercomputing Centre [66].

We have generated a crowd of leaders and followers. Leaders never attended to streams and simply picked their own route. Their internal motivation was set to 1 (see Section 5). In all crowds we set the number of leaders to one in ten. The diversity of the crowd helped to form more diverse streams (Figure 37).

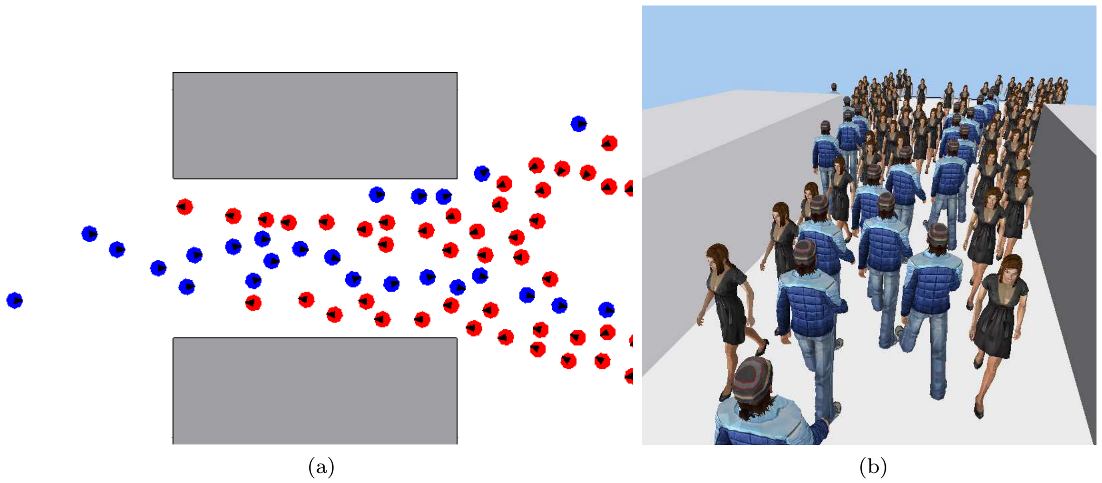


Figure 37: Extended lane formation in the Hermes video rebuild. A 2D and 3D version of the same scenario is shown.

7 Conclusions

Current state-of-the-art micro avoidance algorithms are able to handle either individualistic agents at low densities or groups of agents at high densities. Preferably we would like to achieve an algorithm that can achieve both. Agents should be individualistic entities that cooperate only if needed. At the same time this coordination is essential to crowd motion at high densities. Without any coordination crowds of agents are prone to deadlocks and inefficient movement through space. The essence of lane formation, even at extreme densities, is crucial. Egocentric use of passing opportunities can cause situations to become unresolvable. At low densities, on the other hand, we want agents to move individualistically to prevent overly expending energy through movement. The coordination of agent movement is an area that is still missing in all current micro algorithms.

By interpolating this coordinative behaviour according to density and agent-parameters we can achieve the correct behaviour at most densities. If density is sufficiently high, and the agent has no reason to behave otherwise, it will coordinate its movement. The added energy of trying to move out of a stream outweighs the benefits. As density drops or its incentive rises the agent will start to interpolate between coordination and egocentric behaviour. This interpolation of movement strategies makes sure that the movement is realistic at all densities. The interpolation scheme also allows for different agent motives and goals, giving rise to diverse, individualistic behaviour.

To allow for agent coordination we need an algorithm for high densities. The streams algorithm is an efficient way of creating agent coordination. It is based on the perceived velocity of neighbouring agents and helps the formation of lanes in high densities. The coordination causes agents to remain in formation even if a different movement might be temporarily advantageous. Due to its setup, the streams algorithm is computationally advantageous in high-density environments.

The experiments show that the streams algorithm combined with the strategy interpolation scheme is essential in high-density situations. Low-density situations are not affected by the streams algorithm. We combined an existing micro avoidance algorithm with streams and strategy interpolation. By doing so we have created an micro-avoidance algorithm that is able to handle different densities.

The algorithm by Moussaïd in combination with the vision-based density method gave the best results in our experiments. Computation times were kept low, while still guaranteeing correct behaviour in different density settings. By adding the streams algorithm even high density bi-directional situations can be resolved. Extreme densities, unexpectedly, can still cause trouble.

The flexible setup of agent parameters allows for personalised agent behaviour. By combining different agent profiles we can create a believable, diverse crowd. Each agent within the crowd has a personal goal and can have different personalities.

The streams algorithm helps form and maintain lane formations in high-density, structured crowds. It is a computationally advantageous way to create inter-agent coordination in a crowd at high densities, whilst still maintaining individuality. Unstructured crowds are troublesome for the algorithm. We believe that the streams algorithm is a promising approach that is still in its infancy. Further research should be conducted to extend the streams algorithm to an even wider range of scenarios.

8 Future work

We have proposed two new approaches. Streams help agents coordinate at high densities. Alteration of behaviour dependent on agent factors allows for a widely applicable algorithm. Both approaches are still in their infancy and can be extended further. In this section we will state some further research and open problems that might still be researched.

Uni-directional streams

To prevent to inherent contraction in streams from happening unnecessary, we have limited the use of streams. Only situations containing agents moving in an opposing direction cause streams to be formed. However, two streams moving in the same direction can have a clear influence on each other as well. Preliminary experiments containing two streams crossing at a shallow angle of 30º gave interesting results. Both streams were able to influence each other, by slightly changing their route. If in either stream no new agents arrived, the other stream would slowly return to its original, shortest route. Due to the limitation we set this behaviour now no longer occurs. This behaviour is, however, a real-life phenomenon. Future research should focus on allowing uni-directional streams, whilst still preventing unnecessary contractions in single streams.

Different density measurements

Density has a large impact on the incentive of an agent. In the current implementation this measurement seems to be too strict at low densities. A more lenient formula would allow a higher incentive to improve agent behaviour. At high densities, however, the current 'strictness' is required to make agents coordinate sufficiently. A better density measurement might help improve agent behaviour overall.

Grouping of agents

In the latest research, grouping of agents is becoming more important. Agents rarely travel individually in real life. Our current implementation is lacking the ability to model groups of agents. Clearly, however, group structure has a large impact on stream behaviour.

Several options exist to implement group structure in the stream approach. We could weigh agents belonging to a group more heavily than other agents. This will cause a group of agents to be more likely to make the same choices. Still, agents could be forced apart by streams. A small change in choice might have large effects. Another option would be to measure one incentive value for an entire group of agents. This might make agents more likely to stay together. It is, however, unclear how we should measure this value. A last option would be to add an extra incentive parameter related to the distance to an agents group. This would model the agents desire to stay close to its group. A stream might force it in a different direction, but its desire to stay close is larger.

Composite agents

The concept of composite agents as proposed by Yeh et al. [67] might help stream coordination. In our current approach it is still unclear when an agent should make way and when it should claim ground. A likely approach would be to make a distinction between leaders and followers. Leaders at the head of a group pave the way and should claim ground. Agents following other agents should make way and follow the stream. Composite agents appear to be an easy way to control this inter-agent group-behaviour. If leaders are preceded by a proxy agent, other agents would automatically make way. It is, however, hard to define leaders and followers in real-time.

Troublesome cases

As stated before, there are still a number of cases where the perceived stream velocity is unrealistic (Figure 20). Most cases are forestalled by the interpolation scheme. However, to obtain a stand-alone strong algorithm we should also be able to solve these cases. Obstacle information should be included to make a decision on the stream velocity. Streams should be able to flow towards obstacles, but at a certain point agents should avoid them.

On-the-fly group detection

To be able to properly deal with agent movement and streams, agents should be able to detect groups on-the-fly. If we are able to do this, then we can drastically cut down on computations. When we detect a group, we can make predictions for the entire group at once. This will not only save computations, but increase situation-awareness as well.

Perceiving global movement

This is closely related to detecting groups. All current micro algorithms try to solve crowd flow on local scale. As density increases a more global perspective becomes important. The concept of streams is one of the first to include more global knowledge. Information is conveyed by other agents about the situation ahead. However, if we want agents to move realistically a higher level of abstraction is required. Agents should be able to perceive 'holes' in streams far ahead of time. This type of knowledge would increase crowd flow and enhance situation awareness.

Deadlock resolving

One of the problems we came across while implementing the micro algorithm is deadlocking. In some situations agents can cause deadlocks. Examples include narrow doorways or corridors. Agents are incapable to 'comprehend' situations. As a consequence, they can make movements that may seem illogical. Two agents might decide to move through a doorway at the exact same time. They will prevent each other from ever moving through as agents are incapable to back down. Deadlocking itself can be resolved relatively easily through the use of heuristics. Nevertheless, to obtain realistic agent behaviour, research should be conducted into how humans resolve these situations. Examples can be thought up where agents require extensive explicit inter-agent coordination. A higher level coordination appears to be required.

Following behaviours

In a recent paper by Lemercier et al. [58] studies have been executed as to the following behaviour of agents. Agents appear to adapt their speed to other agents, as well as follow them. This is very closely related to our research. It would be very interesting to see if the results from this paper can be merged with our approach.

Memory

In real-life humans are only able to see the agents ahead. However, humans often keep other agents in mind if they are deemed important. Thus agents also even take into account agents that are not visible any more. Future micro-avoidance algorithms should take this into account. This can be done heuristically, by allowing agents to detect agents behind them as well. An alternative is to actually keep track of this list of dangerous neighbours.

Density detection within edge

Thanks to the density-planning, agents plan their routes around edges that are heavily occupied. However there are cases where density buildup is local. Density-planning is unable to plan around the density-clogs. Locally agents only keep track of a limited number of neighbours. As a result, agents are locally also unable to detect these clogs. Future research should include an approach to detect these avoidable areas locally. The desired route should locally be adjusted to take this into account.

Addenda

A Definitions and Formulas

For the reader's convenience we have stated the most used variables. This should be used as a quick reference when in doubt.

Definitions

Definition 10 (Density) *The density D of an area A is the percentage of free space currently covered by agents within a given, bounded area A.*

$$\begin{aligned} D &= \frac{\left|\bigcap_{i=1}^{n} \Phi_i\right|}{|\Phi_A|}, \textit{where} \\ \Phi_i &= \textit{Total surface covered by agent i;} \\ \left|\bigcap_{i=1}^{n} \Phi_i\right| &= \textit{Size of the combined area of agents } 1 \textit{ through } n\textit{;} \\ |\Phi_A| &= \textit{Total surface of the bounded area A;} \\ n &= \textit{Number of agents within area } A. \end{aligned}$$

Definition 11 (Surface covered by an agent) *The surface covered (Φ_i) by an agent i is the projection of the agent onto the ground surface. A personal space is added.*
Agents prefer to keep a distance to each other [1], which is reflected in the personal space. The combination of both is often approximated by a circle [2, 3, 4].

Terms

Angle of view
The maximum angle within which an agent can observe the environment. The angle is measured as opposed to its movement vector.

Field of view
The total area that is observable by an agent. This area is bounded by the view range and the maximum angle of view.

Line of sight
The unit vector pointing in the direction of movement of the agent.
It equals the movement vector divided by the speed. (i.e. the normalized movement vector)

Movement vector
The vector representing the current velocity of the agent.
This vector represents its direction as well as speed.

Perceived velocity
The virtual velocity of an agent N as perceived by another agent A. This vector represents the contribution that agent N makes to the local stream as perceived by agent A.

Preferred velocity
The preferred velocity that an agent would like to assume. This velocity results from the agent's current goal and maximum speed. The velocity is independent of any moving obstacles such as other agents.

View range
The maximum distance up to which an agent can perceive the environment.

Mathematical variables

α_0	**Angle** between the current movement vector and the preferred direction of an agent as defined by the Indicative Route Method [5].
α_S	**Angle** between the current movement vector and the preferred direction of an agent as defined by the stream.
Φ_i	Occupied **area** on a surface by an object i.
D_A	Local **density** as observed by agent A.
$d_{A,N}$	Euclidean **distance** between agent A and agent N.
$d'_{A,N}$	Relative **distance** between agent A and agent N. Equals the Euclidean distance divided by the view range.
$\mathbf{H_A}$	**Line of sight** of the agent A.
$\mathbf{p_{A,N}}$	**Perceived velocity** of an agent N as perceived by an agent A.
$\mathbf{pos_A}$	Vector describing the **position** of an agent A.
$\|\mathbf{v_A}\|$	Current **speed** of an agent A.
$\mathbf{S_A}$	Local **stream** as perceived by an agent A.
$\mathbf{v_A}$	**Velocity** of an agent A.
d_{max}	**View range** up to which an agent can perceive the environment. (This will be equal for all agents in our thesis, but this is not necessarily the case)

B An overview of the model

In this chapter we will summarize the entire streams model, including the strategy interpolation. In the first section we will give a high level overview describing the combination of the streams algorithm and a micro-avoidance algorithm. The next two sections will give an overview of the different components of the streams model.

B.1 General overview

When computing the velocity for an agent the following steps are taken:

1. An attraction point is determined based on the agent's Indicative Route[5]. By computing the relative position of this attraction point as opposed to the agent's position, we obtain a preferred direction for the agent. (Section 3.2)
   If necessary we update this attraction point to reflect hallways (Section 5.4.1). Instead of taking the regular attraction point we can also select the furthest visible point along the Indicative Route (Section E.1).

2. We compute the agent's incentive and attentiveness. The incentive value will determine to what degree the agent will follow its preferred direction and to what degree it will go along in the stream. If the incentive equals 1 we can skip step 3-5. (Section B.2)

3. We compute the current perceived stream for the agent using the streams algorithm. This will determine a new direction that represents the local stream. The agent should move in this direction if it wants to move along with the stream. (Section B.4)

4. We linearly interpolate between the agents preferred direction from step 1 and the stream direction from step 3. The incentive value is used as the interpolation-factor. The higher the incentive, the more we factor in the agent's preferred direction. We now obtain a direction that incorporates both the agents personal preferred direction as the local crowd direction.

5. We pass the newly obtained direction to the micro-avoidance algorithm to determine the agent's selected direction. By doing so the agent still has collision avoidance options. The agent's attentiveness determines the extent of the available options as well as its view range.

B.2 Determining incentive

The agent's incentive is affected by four factors, namely deviation, local density, time spent and internal motivation. The internal motivation (a) determines a minimum incentive value that an agent will always have. From the other three variables we select the most pressing one, scale it by $1 - a$ and add it to the internal motivation. The incentive value is used to interpolate the agent's personal preferred direction with the direction require by the perceived stream.

- **Incentive**

$$\lambda_A = a + (1-a) * min(max(f_{dev}, f_{dens}, f_t), 1)$$

- **Deviation**

$$f_{dev} = \begin{cases} 0, & min(|\alpha_0 - \alpha_S|, 2\pi - |\alpha_0 - \alpha_S|) < b_{min} \\ b * \frac{min(|\alpha_0 - \alpha_S|, 2\pi - |\alpha_0 - \alpha_S|}{0.25\pi}, & \text{otherwise,} \end{cases}$$

where

$$\begin{aligned} \alpha_0 &= \text{Preferred direction of the agent;} \\ \alpha_S &= \text{Stream direction at the position of the agent;} \\ b_{min} \in [0, \frac{1}{2}\pi] &= \text{Minimum threshold before deviation influences incentive;} \\ b \in [0,1] &= \text{Agent-dependent variable.} \end{aligned}$$

- **Local density**

$$f_{dens} = \begin{cases} 1, & D = 0 \\ \frac{c}{D+c}, & \text{otherwise,} \end{cases}$$

where,

$$\begin{aligned} D \in [0,1] &= \text{Local density at the position of the agent,} \\ c \in [0,1] &= \text{Agent-dependent variable} \end{aligned}$$

- **Time spent**

$$f_t = \begin{cases} 0, & t_{cur} < t_{estFinish} \\ d * \left(\frac{t_{cur} - t_{lastPlan}}{t_{estFinish} - t_{lastPlan}} - 1 \right), & \text{otherwise,} \end{cases}$$

where

$$\begin{aligned} t_{cur} &= \text{Current simulation time;} \\ t_{estFinish} &= \text{Estimated finish-time for the current route;} \\ t_{lastPlan} &= \text{Time of last global route-planning;} \\ d \in [0,1] &= \text{Agent-dependent variable.} \end{aligned}$$

B.3 Determining attentiveness

The attentiveness of an agent determines to what degree it is able to perceive the environment. This variable affects both the view distance and the width of the view cone. The attentiveness is dependent on the agent's incentive. If an agent has a low incentive and is following a stream its attentiveness drops. Attentiveness is also influenced by the relative direction of its neighbours. If an agent moves upstream its attentiveness will rise.

$$attentiveness = max\left(min\left(\gamma * \frac{D_o}{D_a}, 1\right), \lambda * \delta\right),$$

where

$D_a \in [0,1]$ = Local density, when measuring only agents with an *alongside velocity*;

$D_o \in [0,1]$ = Local density, when measuring all other nearby agents;

$\lambda \in [0,1]$ = Incentive of the agent;

$\gamma \in [0,1]$ = Agent-dependent variable;

$\delta \in [0,1]$ = Agent-dependent variable.

Agents are supposed to have an *alongside velocity* if $\hat{\mathbf{v_A}} \cdot \hat{\mathbf{v_B}} \geq 0.5$.

B.4 Computing the local stream

To compute the local stream for an agent A, we select the 5 nearest agents. For these agents we compute the perceived velocity. This velocity is not necessarily equal to the actual velocity. The perceived velocity of an agent N represents the direction that agent A should move in to move along with agent N.

The perceived velocity $\mathbf{p_{A,N}}$ of an agent N by an agent A is defined as:

$$\mathbf{p_{A,N}} = f_{A,N} * (\mathbf{pos_N} \hat{-} \mathbf{pos_A}) * \|\mathbf{v_N}\| + (1 - f_{A,N}) * \mathbf{v_N}$$

where

$(\mathbf{pos_N} \hat{-} \mathbf{pos_A})$ = the normalized vector $(\mathbf{pos_N} - \mathbf{pos_A})$;

$\mathbf{pos_N}$ = the current position of agent N;

$\mathbf{pos_A}$ = the current position of agent A;

$\mathbf{v_N}$ = the velocity of agent N;

$f_{A,N}$ = the interpolation factor for agent N as perceived by agent A;

and $f_{A,N} = D_A * d'_{A,N}$ where,

D_A = the local density at agent A ($0 \leq D_A \leq 1$);

$d'_{A,N}$ = the relative distance ($0 \leq d'_{A,N} \leq 1$) between agent A and agent N. $= \frac{d_{A,N}}{d_{max}}$.

As streams may cross we should not take all agents equally into account. Therefore, each neighbour's perceived velocity is accounted a weighting factor. This weighting factor is dependent on the relative direction of the neighbour.

$$w_{A,B} = \alpha + \left(\frac{(\hat{\mathbf{v_A}} \cdot \hat{\mathbf{v_N}}) + 1}{2} \right)^{\beta} * (1 - \alpha)$$
$$\alpha \in [0, 1]$$
$$\beta \in [0, \infty),$$

where

$\hat{\mathbf{v_A}}$ = unit velocity of the current agent A(=the direction of travel);

$\hat{\mathbf{v_{A,N}}}$ = unit velocity of neighbouring agent N;

$(\hat{\mathbf{v_A}} \cdot \hat{\mathbf{v_N}})$ = dot product of $\hat{\mathbf{v_A}}$and $\hat{\mathbf{v_N}}$.

The local stream is defined as the weighted average of all the selected agents. The local stream $\mathbf{S_A}$ for an agent A is defined as:

$$\mathbf{S_A} = \frac{\sum^{N \in visible} w_{A,N} * \mathbf{p_{A,N}}}{\sum^{B \in visible} w_{A,N}},$$

where

$w_{A,N}$ = the weighting factor for an agent N as perceived by agent A;

$\mathbf{p_{A,N}}$ = the perceived velocity for agent N as perceived by agent A.

The set *visible* consist of the nearest x agents within agent A's field of view.

C Implementation of density-measurements

Several density measurements have been compared to compute the local density for an agent. We will describe these measures in this section and will discuss their advantages and disadvantages. We expected the vision-based approach to give the best results when used in combination with the stream approach. The vision-based approach is the most local approach. As streams are also a local concept, we expect to need a local density measurement as well. We will discuss the density measures from a relatively global to increasingly local measurement.

The most global measurement we have considered is the edge-based density approach. In this approach we store the total amount of space occupied by agents per edge of the medial axis of the ECM. As we can also pre-compute the total surface of each edge[18], it is trivial to compute the average density for an edge. This method has low computational costs. The method is also elegant in its design. Updates of the density measurements by agents can be done in constant time per agent. Densities only need to be updated if their values change. Updates of the density measurements consist of additions and subtractions of the surface area of an individual agent. They only need to be computed if an agent changes edges [29].

Even though this method is computationally cheap, it is not very suitable for our requirements. Edges in the ECM can vary greatly in size (Figure 38), causing unexpected effects when an agent moves to a new edge. Agents who displayed purely group behaviour up till now, might suddenly decide to switch to individual behaviour. The lack of locality of this approach makes it less suitable to combine with the stream-based approach. It might be extremely dense at one end of an edge, and totally empty everywhere else. As density is averaged over the edge, behaviour will be the same all over the edge nevertheless. The edge-based density measure is thus a cheap, but less useful density measurement for our stream-based approach.

A logical next step is measuring density using a grid-based approach. This approach is less informed of the environment and therefore less efficient than the edge-based density measurement. It does, however, still allow constant time updates of local density. The grid-based approach is capable of measuring local density, in contrast to the edge-based approach. Depending on the grid size, local density can be measured independent of the local network structure. This increase in local knowledge makes the density-grid more useful for our requirements. Note

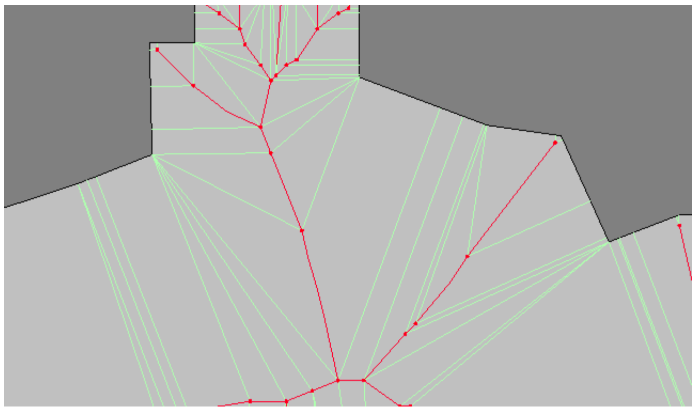


Figure 38: Edges in the Explicit Corridor Map (ECM) can vary greatly in size. Edges in the ECM are represented as red lines.
(Screenshot taken in Pedestrian Dynamics)

[18]In the ECM all closest points are explicitly known. Thus the shape of the area is known and the surface can easily be computed.

that a too small grid size as well as a too large grid size cause trouble with measurements. The density-grid appears on average to be to best compromise, however, as our micro-algorithm already requires a vision-based approach, the extra overhead of measuring density through a vision-based approach are small.

The last method we have considered is a vision-based approach to density measurement. Within the field of view we measure the total area occupied by neighbouring agents. The area covered by an agent consists of the surface covered by the projection of the agent onto the ground surface. We will only account for the part that lies within the field of view. A cheaper alternative is to sum over all agents whose centre of mass is within the field of view. For these agent we will sum the entire area covered even though part of it may be outside of the field of view. This will approximately deliver the same results. On average we would expect to falsely select the same number of agents as that we falsely neglect. When using this approximation it should be noted that density can become larger than one. This should be taken care of by the method.

The main advantage of this method is that the measurement is local. The measured density is the actual density that an agent will encounter. The computed density is valid in the area where the agent will be in the next simulation step. As this density is used to determine how the agent will move, vision-based density is more informed. As the most local measurement of density, this density measurement gives the best results. The computational complexity of computing this density is far larger than the other two approaches described. For each agent we need to recompute this density each turn. We cannot reuse information from the previous simulation step just like the other methods. This method is however the most optimal method when combined with the stream-based approach. If a vision-based algorithm is used to handle collision detection, a large part of these costs already need to be made. In this case the vision-based measurement becomes more attractive. In our experiments we will first compare the different density measurements to determine which is most suited to our streams approach.

D Changes in the vision-based algorithm

To implement the vision-based algorithm by Moussaïd et al. [20] in the software being developed by INCONTROL [30] we had to apply some changes to the basic algorithm. The main update required was an increase in speed as opposed to the original implementation. We have achieved this by efficiently searching through all possible directions of movement for all agent-neighbour pairs. Doing so we have gained an approximate speed-up of a factor 10.[19] The algorithm has also been extended to make agents collide in a more realistic manner. Last but not least some possible extensions for future research have been defined.

D.1 Speeding up

In the basic implementation, as discussed above, many calculations have to be made. For each agent, all possible directions have to be checked against every agent within view. As for each combination a quadratic equation has to be solved, this quickly becomes computationally expensive. Therefore, our objective is to keep these direction-checks to a minimum. We will first compute the point in space where the first collision can ever occur. From this point we compute the angle of movement. By searching only neighbouring angles we can reduce the required search space. We need checks for all angles that result in collisions. Detecting the end of these sets requires one check that does not result into a collision for either side. As soon as we have detected this, no further checks have to be made in the respective direction. Consequently, we have to check at most four angles that do not result in a collision.

Let's assume we have an agent A with a fixed speed and starting point and a neighbouring agent B with a starting point and velocity. We notice that the possible collision angles for agent A form at most two continuous sets. This can visually be perceived by looking at the velocity obstacle [44]. For the neighbouring agent B we create a velocity obstacle (VO). As we know the required speed for agent A, the allowed velocities for A form a set consisting of a circle. At the points where the circle and the velocity obstacle cross, the set of angles becomes colliding or becomes non-colliding. There are exactly three different ways in which this circle and the VO can intersect as seen in Figure 39.

All angles inside the VO cause collisions. These angles form at most two continuous sets. One set is positioned in the direction of the agent B and one is positioned away from agent B. As these sets are continuous, if a subsequent angle does not cause a collision, then all subsequent angles will not cause collisions.

[19] Assuming 15 possible directions per half view cone, which was the required number for realistic behaviour, and a human-sized agent ($r = 0.24m$).

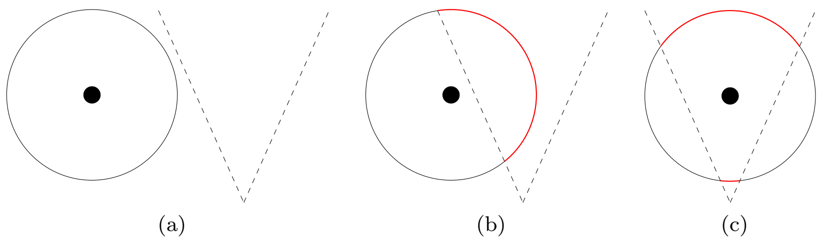


Figure 39: Possible collision angles. Three possible cases exist for the VO. a) In the first case no collisions can occur as all allowed velocities are outside of the VO. b) In case two we have one continuous set of disallowed velocities (drawn in red). c) In the last case, two continuous sets exist.

Algorithm 1 Efficiently calculating collisions

```
function DETERMINEALLCOLLISIONTIMES(agentA, agentB)
    $firstAngle \leftarrow$ FINDFIRSTCOLLISIONANGLE($agentA, agentB$)
    SEARCHAROUNDANGLE($agentA, agentB, firstAngle$)

    $secondAngle \leftarrow$ FINDSECONDCOLLISIONANGLE($agentA, agentB$)
    SEARCHAROUNDANGLE($agentA, agentB, secondAngle$)

    return collisionTimes
end function

function SEARCHAROUNDANGLE(agentA, agentB, angle)
    for $direction = -1; direction \leq 1; direction+ = 2$ do
        $i \leftarrow 1$
        $keepSearching \leftarrow true$

        while $keepSearching$ do
            $collisionTime \leftarrow$ CHECKSINGLEANGLE(
                    $agentA, agentB, angle + direction * i * angleStep$)

            if not collisionOccured or i == maxAngle then
                $keepSearching \leftarrow false$
            end if
        end while
    end for
end function
```

This continuity can be exploited by the algorithm. We compute an initial angle for each set at which both agents will collide. A neighbourhood search will efficiently visit all angles leading to collisions. As soon as an angle is discovered that does not cause a collision, the search can be stopped in this direction.

We determine the angle which will cause both agents to collide at the earliest moment in time. This angle must be part of the first set directed towards agent A. Using the same approach we can also determine the angle that causes both agents to collide at the latest moment in time. It is trivial to see that this angle must be part of the other set. In Algorithm 1 this approach is sketched in pseudo-code.

Given the initial situation as described above, we need to find the earliest moment at which two circles will collide, given an unknown direction for one of the circles. We change the problem in the Minkowski sum of both circles and a point element. This problem results in the same solution.

Let us call our subject agent A, which is the one for which we only know a velocity, and define its shape[20] as the Minkowski sum [68] of both agents. Thus the neighbouring agent B is defined as a point moving along a line with a given speed and starting point. As agent A can move in all possible directions, its furthest reaching point as seen from the starting position of A is described by a circle with a radius based on time.

More specifically $r = r_A + r_B + \Delta t * v$, where v is the given speed of A. Agent A and B will collide at the first possible moment in time when the furthest reaching point of A touches point B (Figure 40). Thus we can describe the set of points $\begin{pmatrix} x \\ y \end{pmatrix}$ that the furthest reaching point

[20]This shape of agent A is, therefore, a circle with radius $r = r_A + r_B$.

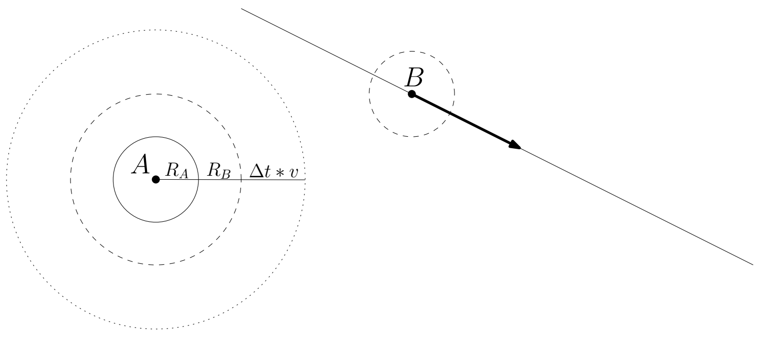


Figure 40: Computing the first collision point. By replacing agent A with the Minkowski sum of both agents, agent B can be reduced to a point. Agent A can move in all directions, but with a given speed. The set of furthest reaching points the new agent A can reach are described by a circle with radius $r_A + r_B + \Delta t * v$. Agent B is described by a point with a linear movement along its velocity vector $\begin{pmatrix} v_x \\ v_y \end{pmatrix}$. When the location of agent B is part of the set of furthest reaching points of agent A, the first possible collision has occurred. Because we now know Δt, we can compute the position of agent B at the time of collision. From this we can deduce the position of agent A and thus its direction.

of A can assume as Equation 4 and point B by Equation 5.

$$(x - x_{O_1})^2 + (y - y_{O_1})^2 = (\Delta t * v + r_1 + r_2)^2 \tag{4}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_{O_2} \\ y_{O_2} \end{pmatrix} + \Delta t * \begin{pmatrix} v_{x_2} \\ v_{y_2} \end{pmatrix} \tag{5}$$

.

For a certain Δt the point B will be part of the set A.[21] So filling in Equation 4 in Equation 5 will give a valid solution.

$$(x_{O_2} + \Delta t * v_{x_2} - x_{O_1})^2 + (y_{O_2} + \Delta t * v_{y_2} - y_{O_1})^2 = (\Delta t * v + r_1 + r_2)^2 \tag{6}$$

If we work this out we get a quadratic formula for Δt which we can solve using the quadratic function [69],

$$\frac{b \pm \sqrt{b^2 - 4ac}}{2a},$$

where

$$a = v_{x_2}^2 + v_{y_2}^2 - v^2;$$
$$b = 2 * (v_{x_2} * (x_{O_2} - x_{O_1}) + v_{y_2} * (y_{O_2} - y_{O_1}) - v * (r_1 + r_2));$$
$$c = (y_{O_2} - y_{O_1})^2 + (x_{O_2} - x_{O_1})^2 - r_1^2 - 2 * r_1 * r_2 - r_2^2.$$

The quadratic formula will return two possible options for Δt. Of these we will select the smallest positive value. Negative values imply a collision in the past and are not meaningful for us. Positive values imply an upcoming collision. Of these collisions we want to know the first, hence, we select the smallest value.

[21] If no collision can ever occur, this is not the case. In this case the quadratic function will have no valid solution. This is not a problem though as we now know that no collision will occur anyhow in this direction.

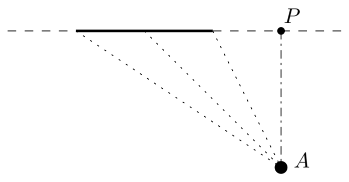


Figure 41: Computing the collision times with a wall-segment. We extend the wall-segment to an infinite line. Now we can compute the nearest point P on this line, as compared to the agents position A. As the angle at P is guaranteed to be perpendicular, we can now quickly compute all collision times.

Having figured out this collision point, it is trivial to figure out the direction and the nearest discrete direction allowed. The same can be done for the latest possible collision moment. In this case, we check the set of points where the Minkowski sum could still be. This set is also described by a circle, but in this case the radius of the circle is described by $\Delta t * v - r_1 - r_2$. Inserting this in Formula 6 and working out the variable Δt will give us another set of values for the quadratic function to calculate the latest possible collision point. Once more we select the smallest positive value

$$\begin{aligned}
a &= v_{x_2}^2 + v_{y_2}^2 - v^2 \\
b &= 2 * (v_{x_2} * (x_{O_2} - x_{O_1}) + v_{y_2} * (y_{O_2} - y_{O_1}) + v * (r_1 + r_2)) \\
c &= (y_{O_2} - y_{O_1})^2 + (x_{O_2} - x_{O_1})^2 - r_1^2 + 2 * r_1 * r_2 - r_2^2
\end{aligned}$$

Efficiently computing walls

We can also speed up computations for walls. We will make the assumption that all walls consist of straight line-segments. This will allow us to quickly compute all collision-times for the directions that will intersect the wall-segment. We make use of a fan, thus reducing the need for a quadratic function to only one instance (see Figure 41).

To compute the collision-time with a wall, we need to compute at what point we will collide with the wall. The naive implementation to compute this is to compute a line-intersection. The line belonging to the wall-segment and the line belonging to the direction need to intersect. However this is computationally heavy, and has to be computed for each direction intersecting the wall segment. Optimally we would like to compute this once and interpolate the values for all other directions intersecting the wall.

Let us assume that the wall segment is infinite. We can now find the nearest point P on this line through the use of the dot-product. We know that the vector towards this closest point will always be perpendicular to the line itself. Therefore the dot product of both must be zero. By solving this equation for P, we can compute its position. We can also determine how far away point P is from our agent's position A. Any direction for the agent that intersects the wall-segment will now give a triangle with one perpendicular angle. Therefore, a quick computation[22] will compute for each angle how far away the collision is (see Equation 7).

$$collissionTime = \frac{d_{A,P}}{cos(\alpha_P - \alpha)} \tag{7}$$

In this formula only α is a variable. Both $d_{A,P}$ and α_P need to be computed only once.

[22]Note that this computation still contains a cosine, which is computationally hard in its own right. Nevertheless, this is outweighed by the saved calculations. If necessary, the cosine function can be replaced by a table of precomputed values.

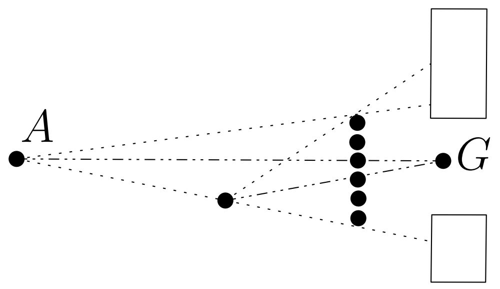


Figure 42: Indecisive avoidance behaviour. An agent A moves towards his goal position G, but finds itself blocked by a group of stationary agents. It picks the right side to pass this group. However in the next step, left appears a better option as it takes longer before the agent will collide with the wall. Making it pick left over right. This successive indecisiveness causes agents to move straight towards the wall of agents.

D.2 Colliding agents due to indecisiveness

To create more realistic collision-avoidance behaviour we extended the algorithm by Moussaïd et al. We address the problem of indecisive agents bumping into their predecessors due to oscillating behaviour.

An important artefact we discovered in the vision-based algorithm is that of indecisive agents. This problem was first noted when trying to let an agent navigate around a stationary group of agents just in front of a small passage. The moving agent is unable to do so, as it can not decide which side is better. It will pick either side and move in that direction. However by doing so, it implicitly creates a better path on the other side of the group of stationary agents. This path deviates from the optimal direction to its goal by just as much, but it takes longer before the agent collides with the wall (see Figure 42). Thus it will switch to the other path. Having switched and crossed the optimal line, the initially picked side will look better once more. This oscillating behaviour keeps the agent on a straight-line collision course, directly heading for the group of stationary agents.

The same behaviour can be noticed between two agents moving in the same direction. If the rear agent has a higher speed, it may want to pass the leading agent. However, in many cases the trailing agent displays oscillating behaviour. It will try to pass left and right subsequently, thereby possibly causing it to run into the back of the leading agent.

To prevent these types of crashes, we have chosen to pick the current speed not solely based on the preferred velocity. We also base the speed on the actual velocity for the agent in the next time step. As velocities are integrated over time, this velocity might be different from the preferred velocity. We compute the allowed speed for both velocities and stick with the most restrictive one. Agents might still be indecisive, but they will not run into their predecessors. We have looked at trying to avoid this indecisive behaviour, but have been unable to come to a satisfactory conclusion. This might still be interesting future research and we will discuss it in the next section.

Minor changes in steering We have also made some minor alterations regarding steering behaviour. First of all we allow agents to turn instantly when they are stationary. We have also chosen to pick the extreme options in case the preferred direction is 'out of sight'. In our algorithm it can occur that the preferred direction is outside of the field of view. In these cases the micro algorithm by Moussaïd may compute an incorrect preferred direction. An example is provided below. We therefore override the algorithm and select the correct extreme direction.

An visual depiction of this example is supplied in Figure 43. Suppose an agent A prefers to move in direction $\mathbf{p}$. Both direction $\mathbf{r}$ and $\mathbf{s}$ will be selected as possible direction. Both angles

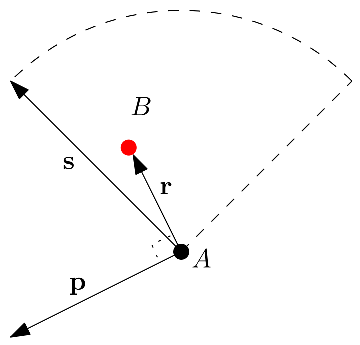


Figure 43: Selecting the incorrect direction. A too large deviation between preferred and tested angles can lead to incorrect selection of directions. In the example agent A prefers direction $\mathbf{p}$. In direction $\mathbf{r}$ it is blocked by an agent B. Nevertheless, agent A will select direction $\mathbf{r}$, even though $\mathbf{s}$ is clearly better. All angles and distances are exaggerated for illustration purposes.

will be compared using the evaluation formula:

$$d(\alpha) = \sqrt{d_{max}^2 + f(\alpha)^2 - 2d_{max}f(\alpha)cos(\alpha_0 - \alpha)}\,.$$

As $\mathbf{r}$ is perpendicular to $\mathbf{p}$, its evaluation function will reduce to $d(\alpha_r) = \sqrt{d_{max}^2 + f(\alpha_r)^2}$.

In this example we will suppose that $f(\alpha_r) = 0.1$ and $f(\alpha_s) = d_{max} = 8$. Also we suppose that $cos(\alpha_0 - \alpha_s) = 0.1$, in other words the angle between p and s almost equals 84 degrees. Clearly direction s is the preferred direction. However, a comparison between both evaluation function will prefer direction $\mathbf{r}$ over $\mathbf{s}$.

$$d(\alpha_r) = \sqrt{8^2 + 0.1^2} \approx \sqrt{64} < \sqrt{(105.2} = \sqrt{8^2 + 8^2 - 2 * 8 * 8 * 0.1} = d(\alpha_s)$$

D.3 Future improvements

Some possibilities for future work are still left. Instead of the discrete vision approach as taken in the original algorithm [20], a continuous approach should be possible. As already described above, we can quite easily determine the boundary angles in between which collisions occur between two agents. The method is based on the VO approach [44]. As in their paper we compute the velocity obstacle, but unlike their paper we have even more information as we know the required speed for our velocity to be picked. The circle of allowable velocities might cross the velocity obstacle in which case collisions can occur for all angles that lie inside the VO. Once again we can compute the intersection points for each boundary surface of the VO with the circle using the quadratic function. This will give all boundaries in continuous space of the collision angles. Some clever heuristic might interpolate in-between these boundaries. We have chosen not to do so, as this method is far more computationally expensive than ours and the discrete approach functioned well enough for our purposes.

Another future improvement which might be researched is how to make agents more decisive. We have tried some different approaches like culling part of the allowable choices after picking a side, but have not reached any satisfactory conclusions. A different approach might be to penalize directions that have been taken recently, giving rise to a sort of biased search.

E Best practices

During our research we have come across several options to increase crowd behaviour in practice. None of these are part of our experiment setup. Most were not selected as although they improve behaviour in practice, they theoretically not advantageous. Others were not created in time to be added in our experiments.

E.1 Looking ahead

We would like to have a solution which is less dependent on the environment. By making an agent pick an attraction point on the Indicative Route (IR) that is further away, fewer local problems occur. We pick an attraction point on the IR that is at the maximum viewing range of an agent. If the point at maximum range is not visible, we will take the furthest visible point along the IR. In the next section an implementation is supplied to compute this point efficiently.

Definition 12 (Furthest attraction point) *The furthest attraction point* $\mathbf{AP_f}$ *for an agent A is the furthest visible point along the indicative route that is visible to the agent A. This point satisfies the following conditions:*

- $\mathbf{AP_f} \in IR$;
- $d_{pos_A,AP_f} \leq d_{max}$;
- $\forall \mathbf{p} \in IR : d_{pos_A,p} \leq d_{pos_A,AP_f}$ *or* $\mathbf{p}$ *is not visible.*

By taking the furthest attraction point we give the agents more global knowledge. Independent of the environment, agents are able to look ahead in their route. This allows the agents to move along a stream even if this is locally not optimal. It also causes agents to pick the right lane before they reach a crossing. Thus avoiding unrealistic chaos on the crossings and improving throughput of agents. Note that the clearance circle might be larger than the maximum viewing range. In these cases it can be decided to use the original attraction point. If the clearance disc is large, the agent is in a wide open space. The viewing range is arguably larger in these cases, thus justifying the use of the original point.

E.1.1 Implementation

It is desirable to give agents more knowledge about the environment. The extent of this knowledge should be independent of the local corridor width. To do so, we need to define the attraction point independent of the local clearance. As discussed we will give the agent an attraction point at a maximum viewing range. Outside the clearance disc, however, obstacles make life hard. In this section we will explain how to pick the furthest visible point along the Indicative Route (IR). We will assume that an agent has a maximal view distance.

From the Explicit Corridor Map (ECM), we can deduct the walls of the corridor up to a maximum view distance d_V. The strength of the ECM is that this can be done relatively easy. From these wall we can deduce the most restricting point on either side ($\mathbf{RP_l}$ and $\mathbf{RP_r}$) (see Figure 44). These point restrict the field of view laterally. They should always be the last points along the relative visible walls.[23] Once we know these points we can quite quickly determine the furthest visible point.

The absolute furthest point $\mathbf{p_f}$ we can reach along the IR is located on a circle with radius d_V. If this point is visible, we are done. We can check if a point is visible by comparing its angle opposed to the movement vector (α_{p_f}). A point is visible if its angle is in the visible range. That is, if it is not blocked from sight by the restrictive points. To be visible its counter-clockwise

[23]A point A along the walls can never be followed by a less restricting point B. Point B would already be restricted from view by point A.

angle (ccw) α_{p_f} must be smaller than the angle α_{RP_l} between the left restrictive point and the movement vector. Similarly, it should be smaller than the angle α_{RP_r} between the right restrictive point and the movement vector. So, we have that

$$\alpha_{RP_r} < \alpha_{p_f} < \alpha_{RP_l},$$

where

$$\begin{aligned}\alpha_{RP_r} &= \text{The ccw angle between the vector } \mathbf{RP_r} - \mathbf{pos_A} \text{and the movement vector } \mathbf{v_A};\\ \alpha_{p_f} &= \text{The ccw angle between the vector } \mathbf{p_f} - \mathbf{pos_A} \text{and the vector } \mathbf{v_A};\\ \alpha_{RP_l} &= \text{The ccw angle between the vector } \mathbf{RP_l} - \mathbf{pos_A} \text{and the vector } \mathbf{v_A}.\end{aligned}$$

If $\mathbf{p_f}$ is not visible, we need a point that is closer to the agent. The furthest visible point is located on the intersection with the line $L = \{(1-t) * \mathbf{RP_l} + t * \mathbf{pos_A} \,|\, t \in \mathbb{R}\}$ or the line $L = \{(1-t) * \mathbf{RP_r} + t * \mathbf{pos_A} \,|\, t \in \mathbb{R}\}$. Starting at $\mathbf{p_f}$, we search backwards to find the first intersection on the IR. The IR is bound to intersect either of these lines somewhere. As this can be arbitrarily close, however, we also check the clearance disc of the agent (see Section 3.2). This way we are guaranteed to find the point $\mathbf{AP_f}$ that is at least as good as the point $\mathbf{AP_{IRM}}$. A graphics representation of this technique is displayed in Figure 44.

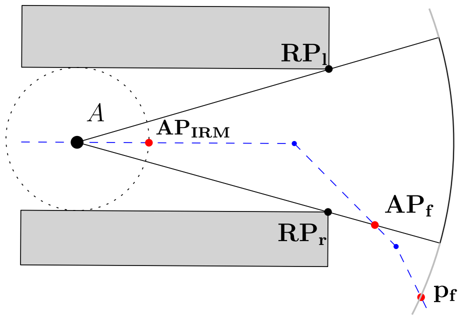


Figure 44: Determining the furthest attraction point $\mathbf{AP_f}$. The Indicative Route is drawn in blue. The maximum view range is drawn in gray. The part of the field of view that is visible is represented in black. $\mathbf{RP_l}$ and $\mathbf{RP_r}$ are the most restricting points. $\mathbf{AP_f}$ is the returned attraction point by the algorithm.

E.2 Local Obstacles

E.2.1 Motivation and definition

Agents plan their global routes through the entire environment. This global planning is taken care of by the ECM [27]. A disadvantage to this approach is global knowledge. If an event occurs that causes a path to be blocked, this will update the global graph of the environment. Even agents far away will immediately know that a path is blocked. They will instantly plan a different route to reach their goal. In some situations this can be undesirable.

Examples of these include police roadblocks and closed doors in an evacuation scenario. In both cases we would expect an agent not to know about these blockades until they have reached them. Depending on the availability of an alternative path, agents should either replan or wait for the situation to be resolved.

We do not want local obstacles to affect the global planning. Therefore we will not change the global graph. Local obstacles are purely local static obstacles. Agents should try to avoid colliding with them and should experience collision forces when in close contact. Local obstacles can have any shape and size. For ease of discussion, however, we will assume them to be concatenated line-segments. An example would be a polygonal object. Our algorithm will not require them to be polygons. Each individual line-segment of a local obstacle can be set as opened or closed. A line-segment can be closed from either side independently.

E.2.2 Local reaction

When an agent approaches a local obstacle it can react in two ways. Either it can globally update its knowledge of the environment or it can locally try to resolve the situation. In the second case, agents will try to keep true to their original route. If possible an agent will avoid the local obstacle. As it reacts in a local fashion, this can not be guaranteed. There are, nevertheless, situations where this behaviour is desired. The most clear example is a local obstacle that fully blocks the only access road. When active, agents should never re-assess their global environment. This would cause the agent to be unable to reach its goal position. The agent will not be able to decide on a course of action. The only acceptable reaction is to ignore the local obstacle on a global planning scale. We can also imagine a controlled throughput, where only a number of agents are allowed to pass per time interval. Agents need to wait until they are allowed to pass and should not replan if the throughput is temporally closed.

Notice that this avoidance is purely a local avoidance. Global avoidance and replanning is not included. As a result, local minima can occur in the search space. In the instances where only a local reaction is required, agents should start waiting at a local minimum, instead of replanning their path. If this behaviour is not desired a global approach should be taken. This will be discussed in the next section.

E.2.3 Global reaction

The concept of local obstacles is that agents have no knowledge about them, unless they have observed them. Agents should only keep the visited local obstacles in mind when picking their route through the environment. This allows agents to discover local obstacles while executing their route. If an agent has discovered a local obstacle, several reactions are possible. The most basic reaction is to keep agents unaware of the local obstacle. If agents can avoid the local obstacle while staying true to their current route they will do so. When no avoidance is possible, agents will gather in front of the local obstacle. Agents might replan, but this does not necessarily cause them to abandon their planned route.

A better option would be to make agents aware of this local obstacle. This can be done as soon as they see it or once they reach it. By making them aware, we allow agents to replan their route to avoid this local obstacle. To allow for this replanning, we need to update their mental map of the environment. This would require to recompute the graph network. As this is expensive if we have to do so for each agent, we would like to prevent this from happening. A solution to the problem would be to calculate the different options for the ECM beforehand and store these. It is sufficient to store the local changes to the ECM. The agent only has to keep track which version is currently in his mental map, for each local possibility.

E.2.4 Implementation

When avoiding local obstacles, we use the functionality already available in the micro-algorithm by Moussaïd. Each line-segment can be seen as a part of a (non-existing) wall. As discussed in Section D.1 we can efficiently compute all collision-times for the respective angles. This will make agents avoid local obstacles.

Even though agents will slow down, collisions can still occur. If an agent has crossed the line, collision forces should be applied. We store the previous position of the agent to detect these cases. A detection is modelled as an intersection test of two line segments. We will also need to determine if the line segment was closed at the side where the agent crossed. This can be computed by taking the cross product between the vector pointing towards the agent and the vector pointing along the local obstacle.

If the agent has just crossed the line segment and it did so from a side that was closed, then we need to take action. In this case, we will reset the agent to its previous position and set its velocity to zero. Please note that this can give undesired results, as agents might be set to a colliding position. Our algorithm does not prevent this. A different option might be to alter the velocity. This would make the agent exit the obstacle in the next simulation step.

Note that some care has to be taken as well if the agent is too close to an obstacle, but has not crossed it yet.

E.3 Composite Agents

E.3.1 Motivation

Composite agents is a concept proposed by Yeh et al. [67]. It is used to efficiently model complex agent interactions. With composite agents, agents can have a virtual proxy-agent. This proxy agent is observed as a regular agent by all other agents. For the current agent and for visualisation purposes, however, no actual agent is present. There are a lot of different social situations in which the concept of composite agents can be used. In this thesis we will mainly be interested in its ability to mimic aggression between agents. This 'aggression' will cause other agents to allot more space to the aggressive agent, even if this causes them to make a larger detour than necessary. We assume that in real life inter-agent communication is a most commonly used way to agree on collision avoidance. Examples might be the hurried businessman trying to catch his train. His simple expression and externalisation of his intentions will make other agent more inclined to avoid him. Another option is a police agent who has to move upstream of a moving group of people. More space will be accorded to him, as he is very decisive to move upstream.

E.3.2 Implementation

We have implemented the aggression representation of composite agents. The more incentive[24] an agent has to follow its own desired route, the more aggressive it will be. If its aggression is high enough, a proxy-agent will be assigned to it. The proxy-agent is placed in such a way that it touches the current agent. It is directed in the preferred direction towards the goal the original agent has. The radius of the proxy-agent is dependent on the aggressiveness of the agent. This implicitly causes the other agents to make way. If an agent has a higher aggressiveness factor, other agents will make way sooner and move aside further.

Problems occurred however when multiple agents tried to move upstream of a large group of oppositely directed agents. All agents moving upstream had a high incentive and thus used proxy-agents. Consequently this allowed each of the agents to follow its preferred route as defined by the Indicative Route Method [5]. This is in contrast with the expected lane formation [1, 4, 22]. Agents join in their effort to move upstream. In other words, we would expect the first agent to make way and the rest to follow. Thus we only allow agents to have a proxy-agent if no other agents in view, moving in the same direction, have a proxy-agent. Preliminary experiments showed that this gave a better result, however, we decided against using composite agents.

Future research might still be executed on the use of composite agents. Composite agents appear an efficient way to model inter-group interaction. Coordination between groups could implicitly be handled by a (or several) proxy agents.

[24]Please refer to Section 5.1 for an extensive discussion of an agent's incentive.

E.4 Resolving deadlocks

Deadlock resolving has not been integrated into our experiments as the goal was to avoid these deadlocks. Resolving deadlocks, while useful in practice, would make results less clear. Also, the approach we take here is a very informal one. While it resolves deadlocks in most occasions it is by no way guaranteed to resolve all deadlocks. We point out to the reader that this a solution to be used in practice, but this is by no means theoretically sound.

In order to resolve deadlocks we first need to detect a deadlock. We assume that only agents at the front of the deadlock should start to resolve it. Other agents should simply move once the deadlock is resolved. We assume that a deadlock can occur if two opposing groups of agents block each other. Note, that this is not necessarily the case and deadlocks of 5 or more agents can be thought of where this is not the case. For each agent we note for a direction if the first collision in this direction is with an opposing agent. If so, we note this direction as 'blocked'.

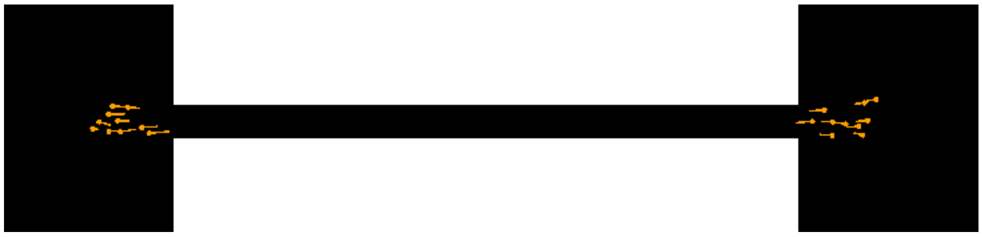
Due to the setup of the micro-avoidance algorithm by Moussaïd et al. agents will try to keep moving for as long as it can. If they come to a full stop and their preferred direction is labelled blocked, we assume a deadlock is occurring. To resolve this deadlock we make the agent more assertive. That is, we half its radius in our computations. This will allow to squeeze through smaller gaps, while not being affected by collision forces. By doing so, we represent the ability of humans to change their diameter by, for example, turning their shoulders. It is important to do the same for the opposing agent as well, as otherwise the opposing agent will be pushed away. This appears to be beneficial, but causes deadlocks to be sustained as agents are pushed along the width of the deadlock. We also enforce a minimum speed of 0.6m/s to represent the agents determination to resolve the deadlock.

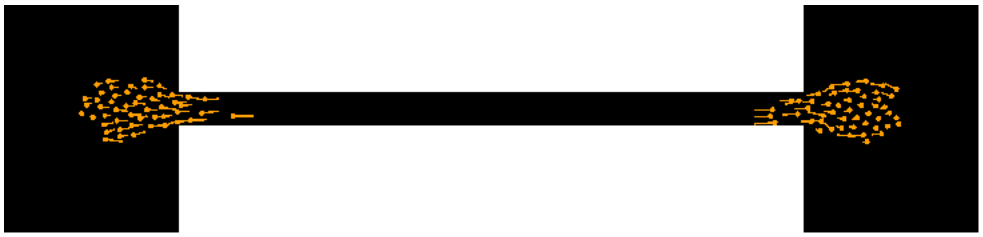
Agents stop being assertive once they have reached a speed of at least 1.2m/s, have reached their maximum speed or are not blocked any more.

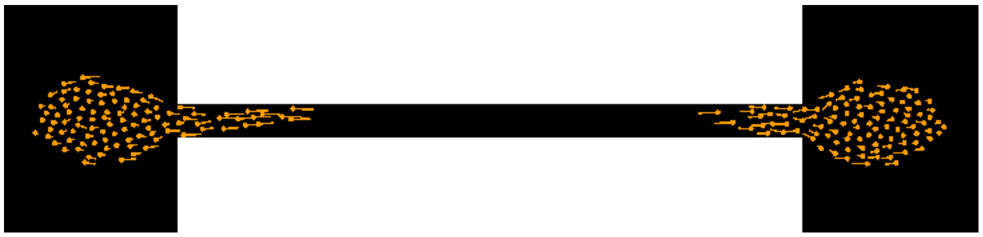
E.5 Multi-core processing

To speed up the algorithm we can run it in parallel. The micro-collision avoidance algorithm including the streams algorithm can be computed simultaneous for all agents. We need to make a distinction between the current velocity and the newly computed velocity. Having done so, however, computations for different agents do not affect each other any more. This multi-core processing helps speed up the algorithm. We were able to run scenarios with thousands of agents with individual goals at several times real-time.

F Depiction of experiments


(a) Narrow-hallway-10


(b) Narrow-hallway-50


(c) Narrow-hallway-100

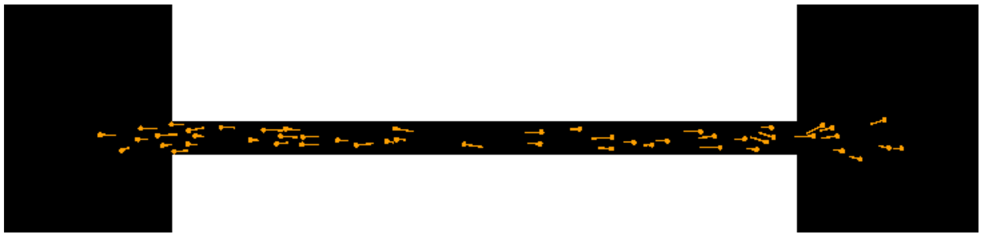
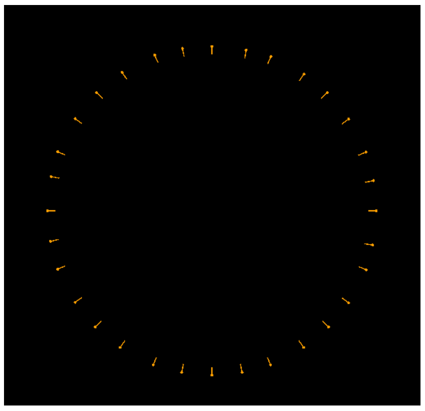
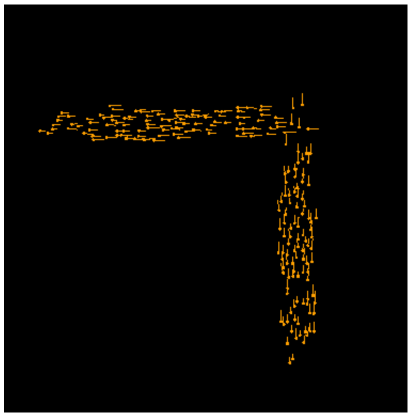
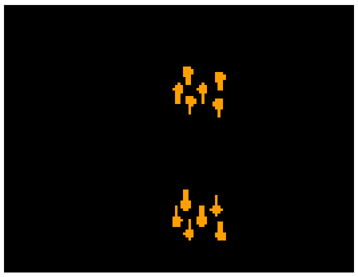

(d) Narrow-hallway-continuous

Figure 45


(a) Circle


(b) Crossing-streams


(c) Oncoming-groups

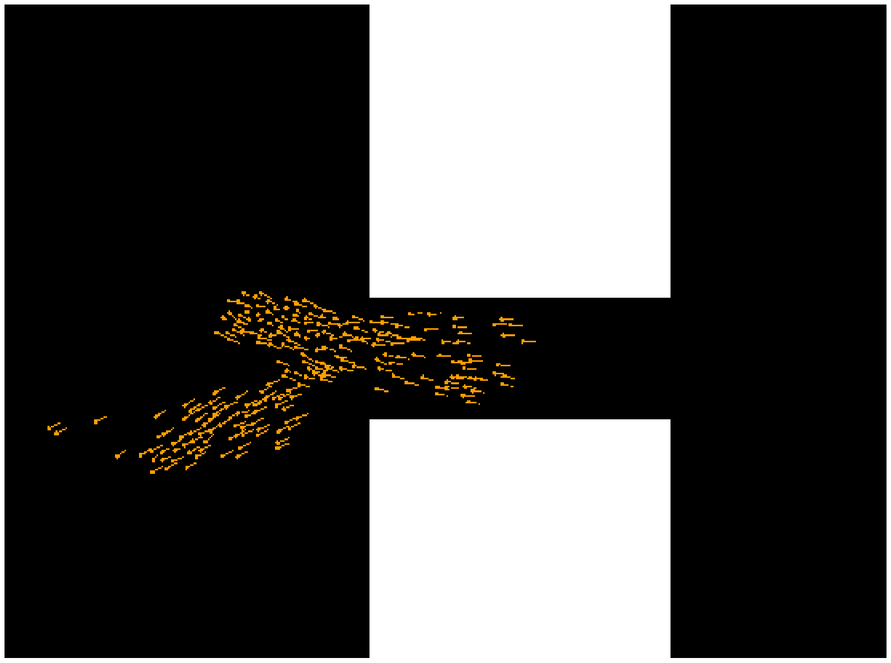
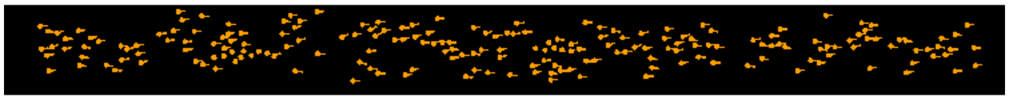
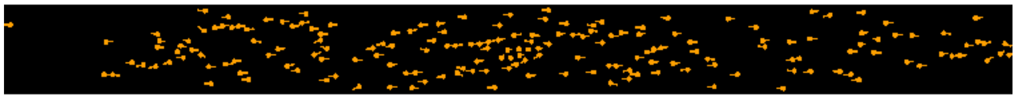

(d) Merging-streams

Figure 46



(a) Hallway-one-way (Steerbench)



(b) Hallway-two-way (Steerbench)

Figure 47

References

[1] D. Helbing and P. Molnár, “Social force model for pedestrian dynamics,” *Physical review e*, vol. 51, 1995.

[2] I. Karamouzas, P. Heil, P. Beek, and M. H. Overmars, “A predictive collision avoidance model for pedestrian simulation,” in *Proceedings of the 2nd International Workshop on Motion in Games*, ser. MIG ’09. Springer-Verlag, 2009, pp. 41–52.

[3] J. van den Berg, M. C. Lin, and D. Manocha, “Reciprocal velocity obstacles for real-time multi-agent navigation,” in *IEEE international conference on robotics and automation.* IEEE, 2008, pp. 1928–1935.

[4] S. J. Guy, J. Chhugani, S. Curtis, P. Dubey, M. C. Lin, and D. Manocha, “Pledestrians: a least-effort approach to crowd simulation,” in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA ’10, 2010, pp. 119–128.

[5] I. Karamouzas, R. Geraerts, and M. H. Overmars, “Indicative routes for path planning and crowd simulation,” in *Proceedings of the 4th International Conference on Foundations of Digital Games*, ser. FDG ’09. New York, NY, USA: ACM, 2009, pp. 113–120.

[6] A. Klar and R. Wegener, “Kinetic traffic flow models,” in *Modeling in Applied Sciences: a kinetic theory approach.* Springer, 2000, ch. 8, pp. 263–316.

[7] B. E. Moore, S. Ali, R. Mehran, and M. Shah, “Visual crowd surveillance through a hydrodynamics lens,” *Commun. ACM*, vol. 54, no. 12, pp. 64–73, 2011.

[8] F. Venuti, L. Bruno, and N. Bellomo, “Crowd dynamics on a moving platform: Mathematical modelling and application to lively footbridges,” *Mathematical and Computer Modelling*, vol. 45, pp. 252–269, 2007.

[9] Boltzmann equation, http://en.wikipedia.org/wiki/Boltzmann_equation, 2012.

[10] S. A. H. AlGadhi and H. S. Mahmassani, “Simulation of crowd behavior and movement fundamental relations and application,” 1991.

[11] I. Karamouzas and M. H. Overmars, “Simulating and evaluating the local behavior of small pedestrian groups,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 3, pp. 394–406, 2012.

[12] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz, “The walking behaviour of pedestrian social groups and its impact on crowd dynamics,” *PLoS ONE*, vol. 5, 2010.

[13] F. Qiu and X. Hu, “Modeling group structures in pedestrian crowd simulation,” 2009.

[14] S. Takahashi, K. Yoshida, T. Kwon, K. H. Lee, J. Lee, and S. Y. Shin, “Spectral-based group formation control,” *Comput. Graph. Forum*, vol. 28, no. 2, pp. 639–648, 2009.

[15] S. Sarmady, F. Haron, and A. Z. Talib, “Modeling groups of pedestrians in least effort crowd movements using cellular automata,” in *Proceedings of the 2009 Third Asia International Conference on Modelling & Simulation*, ser. AMS ’09. IEEE Computer Society, 2009, pp. 520–525.

[16] N. Wijermans, “Understanding crowd behaviour : simulating situated individuals,” Ph.D. dissertation, University of Groningen, 2011.

[17] M. Granovetter, "Threshold models of collective behavior," *American Journal of Sociology*, vol. 83, no. 6, pp. 1420–1443, 1978.

[18] S. J. Guy, S. Kim, M. C. Lin, and D. Manocha, "Simulating heterogeneous crowd behaviors using personality trait theory," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '11. New York, NY, USA: ACM, 2011, pp. 43–52.

[19] A. Lerner, E. Fitusi, Y. Chrysanthou, and D. Cohen-Or, "Fitting behaviors to pedestrian simulations," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '09. ACM, 2009, pp. 199–208.

[20] M. Moussaïd, D. Helbing, and G. Theraulaz, "How simple rules determine pedestrian behavior and crowd disasters," *Proceedings of the National Academy of Sciences*, vol. 108, no. 17, 2011.

[21] J. Ondřej, J. Pettré, A.-H. Olivier, and S. Donikian, "A synthetic-vision based steering approach for crowd simulation," *ACM Trans. Graph.*, vol. 29, 2010.

[22] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," *ACM Trans. Graph.*, vol. 25, pp. 1160–1168, 2006.

[23] R. Narain, A. Golas, S. Curtis, and M. C. Lin, "Aggregate dynamics for dense crowd simulation," *ACM Trans. Graph.*, vol. 28, 2009.

[24] S. Patil, J. van den Berg, S. Curtis, M. C. Lin, and D. Manocha, "Directing crowd simulations using navigation fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 2, pp. 244–254, 2011.

[25] S. Chenney, "Flow tiles," in *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation.* Eurographics Association, 2004, pp. 233–242.

[26] Pedestrian Dynamics, http://www.incontrolsim.com/en/pedestrian-dynamics/pedestrian-dynamics.html, 2012.

[27] W. van Toll, A. F. C. IV, and R. Geraerts, "Navigation meshes for realistic multi-layered environments." IEEE, 2011, pp. 3526–3532.

[28] R. Geraerts, "Planning short paths with clearance using explicit corridors," in *IEEE international conference on robotics and automation*, 2010, pp. 1997–2004.

[29] W. van Toll, A. F. C. IV, and R. Geraerts, "Real-time density-based crowd simulation," in *Computer Animation and Virtual Worlds*, 2012.

[30] INCONTROL Simulation Software, http://www.incontrolsim.com, 2012.

[31] S. M. LaValle, *Planning Algorithms.* Cambridge University Press, 2006.

[32] A* search algorithm, http://en.wikipedia.org/wiki/A*_search_algorithm, 2012.

[33] Dijkstra's algorithm, http://en.wikipedia.org/wiki/Dijkstra's_algorithm, 2012.

[34] J. Pettré, J. Laumond, and D. Thalmann, "A navigation graph for real-time crowd animation on multilayered and uneven terrain," 2005.

[35] B. Yersin, J. Maïm, F. Morini, and D. Thalmann, "Real-time crowd motion planning: Scalable avoidance and group behavior," in *Vis. Comput.*

[36] R. Geraerts and M. H. Overmars, "The corridor map method: a general framework for real-time high-quality path planning," *Comput. Animat. Virtual Worlds*, vol. 18, pp. 107–119, 2007.

[37] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *SIGGRAPH Comput. Graph.*, vol. 21, pp. 25–34, 1987.

[38] W. T. Reeves, "Particle systems - a technique for modeling a class of fuzzy objects," *ACM Trans. Graph.*, vol. 2, pp. 91–108, 1983.

[39] S. Wolfram, *Cellular automata and complexity: collected papers.* Addison-Wesley, 1994.

[40] C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz, "Simulation of pedestrian dynamics using a two-dimensional cellular automaton," *Physica A: Statistical Mechanics and its Applications*, vol. 295, no. 3-4, pp. 507–525, 2001.

[41] V. J. Blue and J. L. Adler, "Cellular automata microsimulation for modeling bi-directional pedestrian walkways," *Transportation Research Part B: Methodological*, vol. 35, no. 3, pp. 293–312, 2001.

[42] H. Yue, H. Hao, X. Chen, and C. Shao, "Simulation of pedestrian flow on square lattice based on cellular automata model," *Physica A: Statistical Mechanics and its Applications*, vol. 384, no. 2, pp. 567–588, 2007.

[43] J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. C. Lin, "Interactive navigation of multiple agents in crowded environments," in *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, ser. I3D '08. New York, NY, USA: ACM, 2008, pp. 139–147.

[44] P. Fiorini and Z. Shillert, "Motion planning in dynamic environments using velocity obstacles," *International Journal of Robotics Research*, vol. 17, pp. 760–772, 1998.

[45] J. E. Cutting, P. M. Vishton, and P. A. Braren, "How we avoid collisions with stationary and with moving obstacles," *Psychological Review*, vol. 102, pp. 627–651, 1995.

[46] J. Pettré, J. Ondřej, A.-H. Olivier, A. Cretual, and S. Donikian, "Experiment-based modeling, simulation and validation of interactions between virtual walkers," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '09. ACM, 2009, pp. 189–198.

[47] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, no. 6803, pp. 487–490, 2000.

[48] D. Helbing, A. Johansson, and H. Z. Al-Abideen, "Dynamics of crowd disasters: An empirical study," *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 75, no. 4, 2007.

[49] A. Johansson, "Data-driven modeling of pedestrian crowds," Ph.D. dissertation, 2009.

[50] J. Fruin, in *Engineering for Crowd Safety*, R. A. Smith and J. F. Dickie, Eds. Elsevier.

[51] "Guide to safety at sports grounds," Department for Culture, Media and Sport, Tech. Rep., 2008.

[52] G. K. Still, "Crowd dynamics," Ph.D. dissertation, University of Warwick, 2000.

[53] R. L. Hughes, "The flow of human crowds," *Annual Review of Fluid Mechanics*, vol. 35, 2003.

[54] R. L. Hughes, “A continuum theory for the flow of pedestrians,” *Transportation Research Part B: Methodological*, vol. 36, 2002.

[55] S. Curtis, S. J. Guy, B. Zafar, and D. Manocha, “Virtual tawaf: A case study in simulating the behavior of dense, heterogeneous crowds.” in *ICCV Workshops.* IEEE, 2011, pp. 128–135.

[56] W. Yu and A. Johansson, “Modeling crowd turbulence by many-particle simulations,” *Physical Review E*, vol. 76, no. 4, 2007.

[57] R. M. Raafat, N. Chater, and C. Frith, “Herding in humans,” *Trends in Cognitive Sciences*, vol. 13, no. 10, pp. 420–428, 2009.

[58] S. Lemercier, A. Jelic, R. Kulpa, J. Hua, J. Fehrenbach, P. Degond, C. Appert-Rolland, S. Donikian, and J. Pettré, “Realistic following behaviors for crowd simulation,” 2012.

[59] F. Aurenhammer, “Voronoi diagrams - a survey of a fundamental geometric data structure,” *ACM computing surveys*, vol. 23, no. 3, pp. 345–405, 1991.

[60] ECM generator, http://www.staff.science.uu.nl/˜gerae101/motion_planning/cm/, 2012.

[61] S. Singh, M. Kapadia, P. Faloutsos, and G. Reinman, “Steerbench: a benchmark suite for evaluating steering behaviors,” *Comput. Animat. Virtual Worlds*, vol. 20, no. 5-6, pp. 533–548, sep 2009.

[62] W. Daamen, “Modelling passenger flows in public transport facilities,” Ph.D. dissertation, Technical University of Delft, 2004.

[63] Steersuite, http://www.magix.ucla.edu/steersuite/, 2012.

[64] I. Karamouzas and M. Overmars, “A velocity-based approach for simulating human collision avoidance,” in *Proceedings of the 10th international conference on Intelligent virtual agents*, ser. IVA’10. Springer-Verlag, 2010, pp. 180–186.

[65] RVO2 library, http://gamma.cs.unc.edu/RVO2/, 2012.

[66] Jülich Supercomputing Centre, http://www2.fz-juelich.de/jsc/appliedmath/ped/, 2012.

[67] H. Yeh, S. Curtis, S. Patil, J. van den Berg, D. Manocha, and M. C. Lin, “Composite agents,” in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA ’08. Eurographics Association, 2008, pp. 39–47.

[68] Minkowski Sum, http://en.wikipedia.org/wiki/Minkowski_addition, 2012.

[69] Quadratic function, http://en.wikipedia.org/wiki/Quadratic_function, 2012.