Continuum Crowds vs Streams

Krijn Thuis

January 29, 2015

Contents

1 Introduction

In our daily lives we encounter many path planning challenges of which many of us are unaware. When we exit a building, we can find our way without consciously planning the path. When trying to model such behavior in computer science we can look at individuals finding their path, at small groups (e.g. a single family) or at large groups (e.g. crowds).

To get a better grip on the problem of path planning, we can subdivide path planning methods into micro, meso, and macro level. When trying to model on a micro level, we try to use perception, motivation, knowledge, beliefs and intentions as input to approximate the behavior of an individual [1] [2] [3]. If we now zoom out to a macroscopic level we can describe the behavior of a group or crowd. You can think of these macroscopic behaviors as flows within a group, lane formation, and stop-and-go waves [4]. At a macroscopic level we can observe that the speed of a group decreases with its density or that people are proned to follow each other [5] [6]. A third level one could picture to oneself, the mesoscopic level, tries to describe behavior of groups on both a microscopic, and a macroscopic level. Mesoscopic methods are gaining popularity, since they combine the advantages of both levels.

In this work we make a comparison between two crowd simulation methods. The first one is a method on a macroscopic level, namely the Continuum Crowds model by Treuille et al. [6]. The second method, a streams-based model as described by Van Goethem et al. [7], is a local method which tries to simulate some macroscopic phenomena and can therefore be seen as a mesoscopic method. Because we want to make a in-depth comparison between those two methods, the main research question of this work is:

> *"In what scenarios will the Continuum Crowds model be more advantageous compared to the streams-based model, and vice versa?"*

To define when a model is 'more advantageous' to use, we will use predefined metrics and evaluate those in different large-scale scenarios. As a reference for such metrics we use the SteerBench framework by Singh et al. [8].

Singh et al. suggest that defined test scenarios, metrics, and evaluations are needed in order to compare two methods in an objective manner. Metrics like number of collisions, path length, and instantaneous acceleration are examples of such metrics. We can use these metrics in different scenarios to answer the following subquestion:

> *"What are the fundamental limitations of both methods?"*

Based on the answer on this subquestion we can also define the advantages of both methods over the other, which will answer our main research question.

2 Related work

Two main methods will be compared, namely the Continuum Crowds model [6] and the Streams model [7]. Both have some close related methods which will be briefly discussed in this section. Crowd simulation methods such as the the Continuum Crowds model [6], describe the behavior of a crowd based on a continuum, like fluid dynamics. Agent-based methods such as the Streams model [7] describe the behavior of a crowd on the level of an individual agent.

One of the first models which tried to describe behavior as a continuum is the Social Force model [1]. This model assumes that every agent is a particle which is attracted and repulsed by several social forces. Because this model was one of the first models to describe crowd behavior, it became a popular way of modeling crowd dynamics. Helbing et al. were for example able to simulate behavior like "escape panic" [9] and the jamming of crowds with large fluctuations [10] (which is the "freezing by heating"-effect).

Since the authors of the Social Force model [1] only used a limited number of social forces, there are many methods which extend this model by defining realistic social forces based on observations in real crowds. One challenging problem in crowd dynamics is a simulation of very dense crowds. Pelechano et al. [11] extended the Social Force model for very dense situations. They subdivide crowd agent motion in three groups: *social force based* approaches, *rule-based* approaches and *cellular automata* approaches. Since *cellular automata* models limit the spatial movement of agents to a grid and *rule-based* methods have difficulty handling high-density crowds because of waiting rules, the HiDAC (*High-Density Autonomous Crowds*) model is presented. This model uses the social force principle combined with components like psychological rules (e.g. panic or impatience) and geometrical rules (e.g. distance or angles). The movement of each agent depends on the desired social force, the repulsive force of obstacles/agents, the desire to move in the same direction and stopping and waiting rules. Compared to traditional Social Force models, the HiDAC model can simulate complex behavior like queuing, pushing, panic or avoiding fallen people in dense crowds.

Other than using social forces, there are also other geometrical approaches. One method which deals with agents as dynamic obstacles instead of static obstacles is described by Van den Berg et al. [12]. In this method each agent considers the *reciprocal velocity obstacle* (RVO) of other agents to determine its new velocity. In theory this method eliminates all inter-agent collisions. The *reciprocal velocity obstacle* of an agent is an obstacle that consists of all allowed velocities of that agent. To imagine this RVO, try to visualize a cone which consists of all possible velocities in the direction of the current velocity of an agent. If we now choose a velocity for each agent which is not contained by a reciprocal velocity obstacle of an other agent, than the velocity will not result in a collision. The advantages over the Social Force based models are the fact that agents take the velocity of other agents into account, the fact that collisions are avoided and the fact that there is a freely available open source implementation of RVO.

Sud et al. [13] note that many agent-based methods use graph searches in order to plan a global path. Sud et al. describe such a method, namely the *MaNG* (*Multiagent Navigation Graph*), which is the union of a discretized 1^{st} and 2^{nd} order Voronoi Diagram. Since the computation of both Voronoi diagrams is non-trivial, this MaNG is constructed using a discretized uniform grid. The MaNG is used to compute the paths of maximum clearance for a set of moving agents and dynamically updates the global paths of agents considering all other agents as obstacles. Using this MaNG, the global path for each separate agent can be determined using the A*-algorithm. A more flexible and recent example of a navigation mesh with an underlying graph that represents the free space is the Explicit Corridor Map method [14], which also uses a Voronoi diagram for path planning with arbitrary clearance from obstacles.

If we now assume we have a *global path* and a way of calculating the velocities for each agent in each timestep, we still need to address the computational limitations of the CPU (since we are dealing with large numbers of agents). An example of a method that

efficiently deals with many local interactions between agents is the PSCrowds model by Reynolds [15]. To efficiently compute interactions between agents in very large crowds, Reynolds proposes the PSCrowds model ([15]) which limits particles (or agents) to local interactions in combination with spatial hashing. Using this hashing, the interaction between particles can be considered an $O(n)$ problem or a barely quadratic problem. The authors [15] furthermore point out that each job in the simulation step needs to be independent (and preferably order-independent) to be executed in parallel. The PSCrowd model uses a lattice- and bucket-structure for local interactions, which is simular to grid- and cell-structure in the Continuum Crowds model.

Since the main goal of this work is to compare the Continuum Crowds model [6] with the Streams model [7], the upcoming two subsections will contain some closely related methods to either the Continuum Crowds model or the Streams model. For the Continuum Crowds model two flow-based methods will be treated that can be seen as an extension on the method by Treuille et al. For the Streams model there are no current extensions since the method is rather new, and, therefore some similar agent-based methods will be discussed. This does not mean that the aforementioned methods do not fall into these categories, but the discussed models in the upcoming two sections are more close-related compared to the aforementioned models.

2.1 Flow-based methods

The first method which will be discussed is the model by Narain et al. [16]. This method addresses the inter-agent dynamics in very dense crowds, which means the partly loss of individual freedom of motion in such crowds. Similar to the Continuum Crowds method the model describes the crowd motion as the flow of one single aggregate system, but is more suitable for crowds where people are densely packed.

The method consists of three large parts, namely the *global path planning* resulting in a preferred velocity, the conversion to a *continuous representation* resulting in a local density/velocity and solving a *unilateral incompressibility constraint* resulting in the actual velocity. The *first step*, the global path planning, is not part of the method itself, but can be selected freely. In their implementation, they use road-maps, as described by Sud et al. [13]. The authors mention that any other path planning method (like the continuum path planner of Treuille et al. [6]) can be used instead. The *second step* is fairly similar to the discretization step of the Continuum Crowds model, where a uniform grid is constructed. The density for each grid cell is the sum of the mass of all agents and the velocity for each grid cell is the sum of all agent velocities. In final or *third step* the unilateral incompressibility constraint (*UIC*) is solved, which is a macroscopic way of inter-agent collision avoidance. This UIC is a hybrid of the fact that a crowd is neither purely incompressible, nor purely compressible. To achieve collision avoidance in dense crowds, the UIC takes a maximum density and a minimum distance as an input and tries to prevent from converging to a higher maximal density by adding pressure. After solving this UIC, the method interpolates between the continuum velocity of the grid and the agents' individual preferred velocity, followed by a pairwise separation force.

Another method which can be seen as an extension to the Continuum Crowds model is the method by Patil et al. [5]. The main contribution of this paper is the addition of user-specified, either by hand or by video footage, *guidance fields* to direct agents. The method is, like the Continuum Crowds method, a goal-directed method for multiple agents. The method consists of two main parts, the determination of the *guidance fields* and *navigation fields* (macroscopic) and solving the *local behavior* (microscopic).

The *first step* is to define the navigation fields and guidance fields, which are represented by unit vectors pointing in a certain direction. The guidance fields are distinct for every group of agents. Patil et al. present many ways to construct the *guidance fields*, like analyzing real-life video sequences, the use of sketch-based interfaces or blending a guidance

field from multiple inputs. When the guidance fields are described, the *navigation fields* are constructed in the form of goal-directed vectors which are almost smooth and without local minima. For simplicity reasons the authors propose the use of a uniform grid to construct such a vector field. To compute those goal-directed vectors, they start by labeling each cell as obstacle or free space. The authors use an interpolation based version of Dijkstra's algorithm [17]; the goal position has a cost of zero and the algorithm propagates while storing the back-pointer for each grid cell. When the authors use this algorithm to propagate the grid, the algorithm will give the shortest path (similar to the Fast Marching algorithm [18] in the Continuum Crowds model) if no guidance fields are present in the simulation.

After the computation of the *navigation field* and the preferred velocity, which is determined by the bilinear interpolation between the four closest neighbor vectors, the *second step* is the local collision avoidance which is again not part of the method by Patil et al. The authors state that any local collision avoidance method can be used. Two local collision avoidance methods that the authors mention are the Social Force Model and the Reciprocal Velocity Obstacles (RVO), which are mentioned earlier this section.

Both methods [5][16] run at interactive rates with a few hundred to a few thousand agents and can thus be considered real-time. Furthermore, if the grid does not need an update, both methods are *nearly* linear in the number of agents per simulation step, since the local collision avoidance step for both methods discretize the free-space and thus results in a nearly constant number of nearby agents.

2.2 Agent-based methods

Since the method by Van Goethem et al. [7] is a recent method and there are no methods that expand on this method, this section will contain several closely related methods. The main contributions of the *Streams* model [7] are the determination of the crowd density and the perceived stream velocity using the *field of view* of an agent in combination with the use of an *incentive* which interpolates between the individual prefered velocity and the perceived stream velocity. These steps take place inbetween the global path planning and the local collision avoidance (more about the Streams model can be found in Section 3).

The aforementioned models, like the Continuum Crowds method, are flow-based and are well-suited for dense crowds with only one goal per group of agents. Flow-based methods are however not optimal for motion in low-density settings, where individual agents all have different goals. Agent-based models do not describe motion on a macroscopic level, but rather describe the motion for every individual agent.

Most agent-based methods have a few components in common, namely the dependency of speed on density and the principle of least effort. Lemencier et al. [4] found that there is a dependency between the density and the velocity in crowds. They observed that velocity decreases when density increases, which is consistent with earlier research [19]. Other observations, like the principle of least effort [20] or the minimization of kinetic energy, are widely accepted as guidelines for an agent-based method.

Apart from those observations, some methods try to use the concepts of certain human senses. There are vision-based models like the method by Ondřej et al. [21] or the method by Moussaid et al. [3], where the field of view is a cone with its origin at the agent's current position centered around the agent's current velocity. Recent work by Kountouriotis et al. [22] suggests that a three-layer model can be used. In this model, people individually sense the environment in the first layer after hearing a sound or visually observing panic of surrounding agents. The remaining two layers are the strategic layer, where the decision making is done based on those observations, and a movement layer, which results in the actual movement.

3 Methods

To create a deeper understanding of the limitations and the advantages of both methods, this section will contain a concise description of both methods. We will start with the Continuum Crowds method in Section 3.1 and then explain the Streams model in Section 3.2.

3.1 Continuum Crowds

The Continuum Crowds model by Treuille et al. [6] is a flow-based method that describes the behavior of a crowd based on a continuum. The first assumption the authors make is that every group tries to reach a geographical goal which should be given. Another assumption is that agents in a group will try to reach this geographical goal with the maximum speed possible. Because real people often prefer certain paths or areas, the authors also assume a discomfort field that defines the desirability of points in the free space. Based on the former three assumptions the authors assume that the speed field (f), the discomfort field (g), and the goal location are fixed. This means that an agent will pick a path P minimizing

$$\underbrace{\alpha \int_P 1ds}_{\text{Path Length}} + \underbrace{\beta \int_P 1dt}_{\text{Time}} + \underbrace{\gamma \int_P gdt}_{\text{Discomfort}} .$$

Using the three weights, α, β and γ, the behavior of agents can be changed. One could give the weight of the discomfort field, γ, a higher value, which will make agents more prone to evade undesired cells. To reduce the amount time spend to the destination the time weight (β) can be increased or to reduce the length of the path we could use the weight for the path length (α). The above formula can be simplified (since $ds = fdt$ where f is the speed) to the equation

$$\int_P C_{a\to b} ds, \quad where \quad C_{a\to b} \equiv (\frac{\alpha f_{a\to b} + \beta + \gamma g_{a\to b}}{f_{a\to b}}).$$

The authors use this equation to compute the *Unit Cost Field* from cell a to cell b. Using this Unit Cost Field a potential dynamic field is constructed to be able to calculate an optimal path for every agent in a group. The velocity with which the agent will traverse this path is dependent on the density and the velocity of the crowd. In dense situations the velocity of other agents will dominate the speed of an agent, but in areas where the crowd is not dense the speed of an agent is dominated by the topographic speed.

The algorithm itself consists of four steps, which will be discussed in this paragraph. As input the agents should be given and the space should be discretized, resulting in a grid. The *first step* is the determination of the velocity and the density for each cell of this grid. For each agent we add density and velocity to the four neighbor cells using a bilinear splatting technique. This means that each of the four neighbor cells is affected by an agent according to the distance between the cell and the agent. This way the closest cell gets more density added compared to the other three neighbor cells. Simultaneously we calculate the *average velocity field* which scales each agents density by its velocity. This way an indication of the overall speed and density is generated.

The *second step* is the computation of the unit cost for each cell. This can easily be done by solving the above equation for each cell, where g is the discomfort and f is an interpolation between the *topographic speed* (largely dependent on the slope) and the *flow speed* (largely dependent on the density). The weights are assumed input and we have the flow speed f and the discomfort g for each cell. This way we can calculate the unit cost C for each cell.

The *third step*, which is the most time consuming, is the construction of the *dynamic potential field*. In this step the goal cell for a group has cost *zero* and using a Fast Marching Algorithm [18] an approximation of the gradient can be calculated for each cell. Starting

at the goal location this algorithm adds the four bordering cells to the CANDIDATE cells. Next the algorithm chooses the cell with the lowest unit cost and adds this cell to the KNOWN cells. The algorithm propagates over all cells this way.

Suppose the algorithm has arrived at cell M, and an approximation of the gradient

$$|| \bigtriangledown \phi f(x)|| = C$$

is needed, where ϕ is the potential function which is known at the goal location ($\phi = 0$).

Both, for the two horizontal neighbor cells and the two vertical neighbor cells, a finite difference approximation to $|| \bigtriangledown \phi f(x)|| = C$ can be done by solving

$$\left(\frac{\phi_M - \phi_{m_x}}{C_{M \to m_x}}\right)^2 + \left(\frac{\phi_M - \phi_{m_y}}{C_{M \to m_y}}\right)^2 = 1,$$

for the larger solution to ϕ_M. In this quadratic equation, ϕ_{m_x} and ϕ_{m_y} are obtained by solving

$$m_x = \underset{i \in \{W,E\}}{\arg\min} \{\phi_i + C_{M \to i}\}, \qquad \textit{and}, \qquad m_y = \underset{i \in \{N,S\}}{\arg\min} \{\phi_i + C_{M \to i}\},$$

where ϕ_i is the potential to a neighbor cell i and $C_{M \to i}$ is the unit cost from cell M to a neighbor cell i. Once ϕ_M is computed, the difference with the neighboring grid cells is taken in the upwind direction giving us $\bigtriangledown \phi$.

In the *fourth step* we can use this approximation of the gradient to determine a displacement velocity using an Euler integration. To ensure that no two agents will collide, the authors propose to use a pair-wise minimum distance force for each agent. This can easily be done by iterating over all pairs and add a force based on the distance of a pair.

3.2 Streams Model

The Streams model by Van Goethem et al. [7] is an agent-based method that describes the behavior of a crowd on an individual level. The authors assume that every agent has his own *field of view* (FOV). The main assumption is that real people mainly navigate using visual input instead of global knowledge of the crowd, which is a fundamental difference from the Continuum Crowds model by Treuille et al. [6]. This FOV has a viewing angle of $\phi = 180°$ degrees with some look-ahead distance d_{max}. The model interpolates between individual and crowd velocity depending on local information. Streams can be seen as flows of agents that actively coordinate by aligning their paths or following each other, that can result in energy-efficient paths in high densities.

The algorithm can be subdivided in five steps of which three are part of the Streams model and two are independent to the Streams model. The *first step* is the step in which the individual velocity is calculated. This is one of the two steps which are independent of the Streams model and can be done by any global path planning method.

In the *second step* the local density has to be determined. Van Goethem et al. tried three different approaches for the calculation of density for the Streams model, namely an *Explicit Corridor Map* (ECM) [14], a grid and a FOV. The Explicit Corridor Map is computationally efficient, but since the ECM cells can vary in size and span, agents might 'experience' high densities, while the density is far away from the actual position of the agent. Using a uniform grid, the cells become equal in size and therefore fit better in this local streams model. However, the computation time is largely dependent on the grid size, and the size of the grid cells influences the density information similar to the ECM-approach. The use of a FOV overcomes this problem of local density information, since each agent has its own field of view with its own local density information.

To define the density the authors use

$$\rho := min\left(\frac{3}{\triangle(FOV)}\sum_{N\in\aleph}\triangle(N), 1 \quad\right),$$

where $\triangle(FOV)$ is the total area of the field of view, $\triangle(N)$ is the area occupied by agent N and $\aleph$ are all agents that have their disc center inside a certain FOV. As can be seen in this formula the authors assume that a one third occupation is already considered a high density situation.

The *third step* is the computation of the perceived stream in the FOV of each agent. This is an interpolation between a *follow strategy* and an *alignment strategy*, based on the relative distance between agents. With the follow strategy agent A will be attracted to the location of agent B, while with the alignment strategy agent A will align with agent B. The *perceived stream* of agent A depends on the velocity v_B of agent B. The perceived velocity $v_{per(A,B)}$ is defined as an interpolation between v_B and the normalized vector $(x_b - x_a)$ scaled by the length of the velocity $(v_{dir(A,B)})$. To compute the perceived velocity factor $f_{A,B} = \rho \cdot d_{A,B}$ with $\rho \in [0, 1]$ is used to interpolate as follows:

$$v_{per(A,B)} := f_{A,B} \cdot v_{dir(A,B)} + (1 - f_{A,B}) \cdot v_B.$$

In short, this formula makes agents pursue a follow strategy when another agent is on the edge of the view-distance, and, makes agents pursue an alignment strategy when very close to another agent. The actual perceived local stream velocity v_{stream} is defined as:

$$v_{stream} := \frac{s}{||\sum_{N\in\aleph_5} v_{per(A,N)}||} \cdot \sum_{N\in\aleph_5} v_{per(A,N)}, \quad with, \qquad s := \frac{1}{|\aleph_5|} \cdot \sum_{N\in\aleph_5} ||v_{per(A,N)||}.$$

After the perceived stream and the individual velocity are defined, in the *fourth step* an interpolation between those can be calculated. The authors use this so called *incentive* to interpolate between both velocities. The incentive is defined by four factors: internal motivation γ, deviation ϕ from its preferred velocity, perceived local density ρ and the total time spent τ. When the incentive is equal to *zero* the agent will follow the streams and when the incentive is *one* the agent will use its preferred velocity. Apart from the internal motivation γ only the dominant of the remaining factors will have impact on the incentive. Furthermore the density ρ will drop rapidly when dropping to zero using $(1 - \rho)^3$. The deviation factor ϕ and the time spent τ are

$$\phi := min\left(max\left(\frac{\phi_{dev} - \phi_{min}}{\phi_{min}}, 0\right), 1\right), \quad and, \qquad \tau := min\left(max\left(\frac{\tau_{spent} - \tau_{exp}}{\tau_{exp}}, 0\right), 1\right).$$

The incentive can λ can now be defined as follows:

$$\lambda := \gamma + (1 - \gamma) \cdot max(\phi, (1 - \rho)^3, \tau).$$

The last or *fifth step* is to solve for local collision avoidance. This can be done by any local collision avoidance method and is the second part which is independent to the Streams model. The authors use methods like a vision-based method [3], an Optimal Reciprocal Collision-Avoidance method [12], and a predictive collision avoidance method [23].

4 Experimental set up

To answer the main research question, a set of scenarios and metrics must be defined. The metrics chosen in this work are obtained from the scenarios used in the papers of both models [6][7]. The metrics used to compare both models are based on the proposed metrics of the Steerbench framework [8]. In this section, the chosen scenarios and metrics are presented.

4.1 Scenarios

Based on the scenarios used in both papers there are six scenarios selected:

> ***Eight classrooms scenario**: The classroom scenario was chosen because it is used in by Treuille et al. The agents start in one classroom and have a goal position in an opposite classroom. See Figure 1. The scenario contains eight groups of 100 agents.*
>
> ***Two opposing groups scenario**: The scenario where two opposing groups walk into each other was chosen because it is used in by both Treuille et al. and Van Goethem et al. In this scenario two rather large groups try to swap positions. See Figure 2. The scenario contains two groups of 100 agents.*
>
> ***Two squeezing groups scenario**: The scenario of two groups squeezing through a narrow alley is chosen because it is used by Van Goethem et al. In this scenario two rather large groups of agents will be squeezed through an allay. See Figure 3. The scenario contains two groups of 100 agents.*
>
> ***Two perpendicular groups scenario**: The scenarios where two groups try to move in perpendicular direction is chosen because it is used by Van Goethem et al. In this scenario two large groups will interact at perpendicular moving directions. See Figure 4. The scenario contains two groups of 200 agents.*
>
> ***Four crossing groups scenario**: The scenario of four groups crossing is used by Treuille et al. and is also chosen for the comparison of both methods. In this scenario four rather large groups will move from the four corners towards the opposite corner. See Figure 5. The scenario contains four groups of 100 agents.*
>
> ***Military scenario**: The military scenario is used by Van Goethem et al. to measure running times of the Streams model. In this scenario we will simulated one large group of agents walking through a military environment. See Figure 6. The scenario contains one group of 200 agents.*

The six scenarios are visualized on the next page, where the square of circles represents a group of agents and the gray polygons are obstacles.

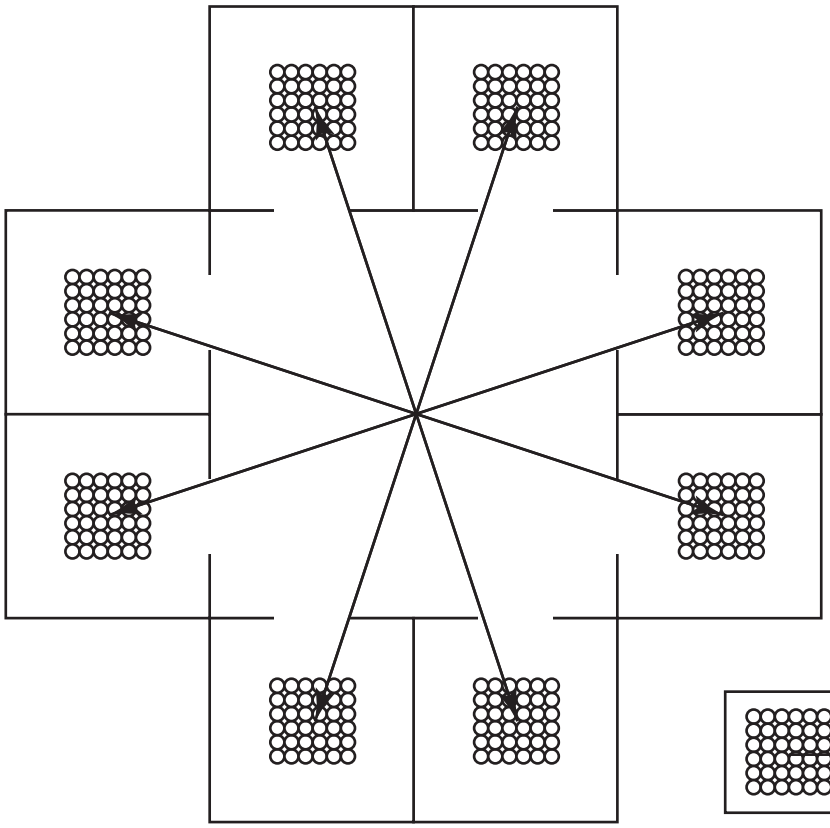

Figure 1: Eight classrooms

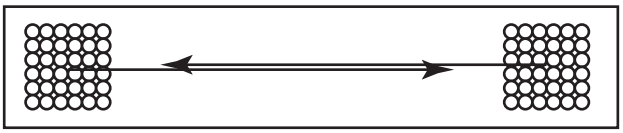

Figure 2: Two opposing groups

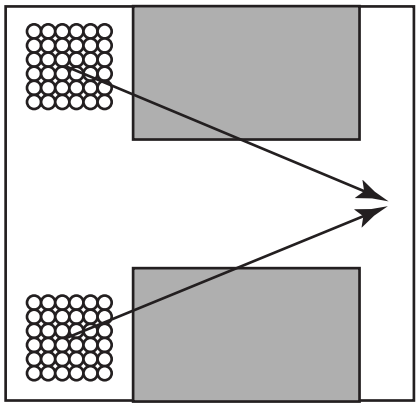

Figure 3: Two squeezing groups

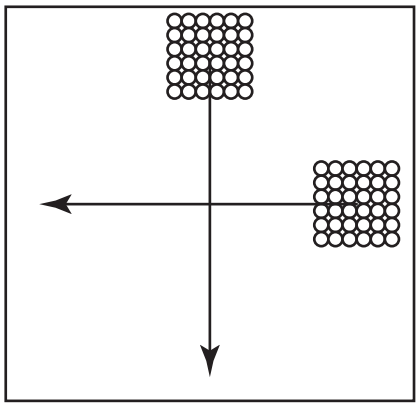

Figure 4: Two perpendicular groups

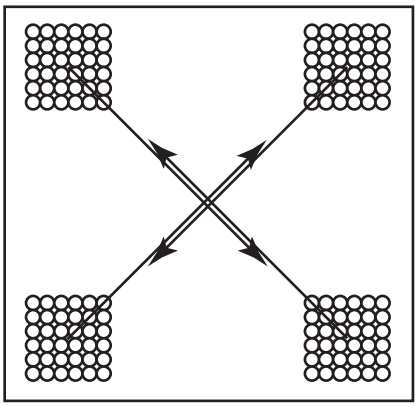

Figure 5: Four crossing groups

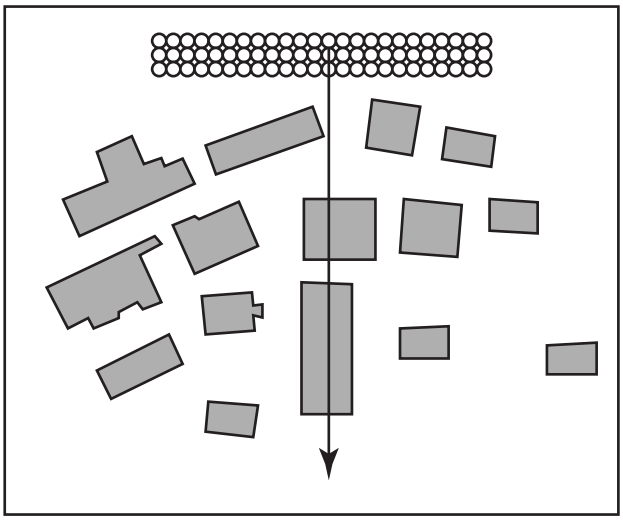

Figure 6: One groups in a military environment

4.2 Metrics

Based on the metrics of the Steerbench suite a set of metrics was choosen. The following table shows the metrics that are used to compare the Streams and Continuum Crowds model:

Reached goal	n	The number of agents that reached the goal within 2 minutes
Collisions	n	The number of collisions
Time to goal	t	The time it takes to reach the goal
Kinetic energy	mv^2	The kinetic energy which is needed to reach the goal
Degrees turned	deg	The total number of degrees turned
Maximum angular speed	deg/t	Maximum instantaneous turn speed
Angular speed change	n	Number of time the angular speed changes significantly
Distance traveled	s	Total distance traveled by an agent
Speed change	v	Total speed change
Maximum speed change	a	Maximum instantaneous speed change
Significant speed change	n	Number of times the speed changes significantly
Average speed	v	Average speed of agents ($v_{preferred} = 1.4$)

Table 1: Chosen metrics used to compare both methods.

We use these metrics to compare different elements of both models. The number of agents that *reached the goal* within a certain time indicates deadlocks within a method. The number of *collisions* indicates poor behaviors or poor local collision avoidance. To measure time efficiency we keep track of the *time to the goal* and we use the *kinetic energy* to measure the efficiency of the effort. We try to describe the effort of turning using the total *degrees turned*, the *maximum angular speed* and the number of time the *angular speed change*s significantly. If the maximum angular speed and the angular speed change are small, this could indicate that the agents turn very smoothly. The path length can be measured by the total *distance traveled*. Just like the measurement of the turning effort we will also keep track of the speed change effort using the total *speed change*, the *maximum speed change* and the number of times the *speed changes significantly*. Just like the metrics for the turning effort, the maximum speed change and the times an agents changes its speed significantly says something about the smoothness of the movement. Finally, we can compare the desired speed of the methods with the *average speed* to see how close the agents are to their preferred speed.

5 Experimental results

Both the Continuum Crowds model and the Streams model were implemented in a framework which is based on the *Explicit Corridor Map*, which is introduced by Geraerts et al. [14]. All experiments have been conducted on a laptop computer running on Windows 7 with a 2.5GHz Intel Core i5-2450M, 8GB RAM DDR3 and a NVIDIA GeForce GT 540M with 2 GB dedicated DDR3. For fair comparison only one CPU core was used. For the Streams model the Explicit Corridor Map with the Indicative Route Method [23] is used for the path planning. For the Continuum Crowds model some educated guesses were done for the calculation of the Unit Cost (see Section 3.1), because this information was missing in the original paper. For the speed the interpolation between the topographical speed and the flow speed were done with a minimum density of 0.4 and a maximum density of 0.9 ($\lambda = 1.0$). For all three the weights of the unit cost field a value of 1.0 was used. The terrain height (used for the topographical speed for the Continuum Crowds model) is neglected and not used for the comparison. For the Streams model the internal motivation is equal for all agents (in this case $\gamma = 0$) and the maximum number of agents processed in the FOV is 5 (as mentioned in the paper [7]). All experiments we conducted multiple times, but only one, which are the most optimal results in that scenario, is shown in the tables. All metrics are the average of all agents in that scenario, except for the number of agents that reached the goal within the time limit, since this is the total number of agents that reached the goal.

The results of the six scenarios discussed in Section 4.1 are shown in Tables 2-7. Using the metrics described in Section 4.2 we marked the value that is considered best using a simple two-tailed t-test over the data of all individual agents in a scenario. The data of the two methods is considered significantly different when $p < 0.01$. For all collected data the sample mean $\bar{x}$ and the sample standard deviation s of the agent population is put down in Table 2-7. See Section 4.2 for more detailed information about the metrics used.

When observing the six difference results, there are several noticeable differences. The *first* observation is that the Streams model, on average, uses the least kinetic energy, because this model results in less speed changes and less angular speed changes. The *second* notable observation is that you would expect that the *Time to goal* is always lowest for the Streams model, since the kinetic energy is lower. This observation however does not always hold, for example in the classroom and the four crossing groups scenarios. The reason why the *Time to goal* is lower for the Continuum Crowds in those two scenarios is the higher average speed, which is closer to the preferred maximum speed ($v_{pref} = 1.4$).

One big footnote with these results is that the Continuum Crowds method is greatly dependent on the grid size and the pair-wise collision force. In Table 7, for example, we can observe that a grid of 240x240 scores better on the *Time to goal* and the *Kinetic energy* compared to the Streams model. Furthermore the Streams model has significantly less collisions in most scenarios, possibly because it does not use a separation force.

Eight classrooms scenario

	Streams model		Continuum Crowds	
	$\bar{x}$	s	$\bar{x}$	s
Reached goal (n)	670,0		800,0	
Collisions (n)	39,4	26,4	37,6	12,0
Time to goal (t)	108,6	11,0	**60,4**	**5,6**
Kinetic energy (mv^2)	**82,6**	**7,2**	98,5	6,5
Degrees turned (deg)	**0,96**	**0,44**	1,05	0,56
Maximum angular speed (deg/t)	3,45	1,62	3,27	1,65
Angular Speed change (v) (n)	**0,57**	**0,72**	0,86	1,40
Distance traveled (s)	910,0	73,1	**797,7**	**53,9**
Speed change (v)	**15,7**	**3,7**	41,6	11,6
Maximum speed change (a) (v)	**0,28**	**0,08**	0,59	0,21
Significant speed change (n) (v)	0,00	0,00	0,00	0,00
Average speed (v)	0,84	0,09	**1,32**	**0,06**

Table 2: Eight groups of 100 agents moving to an opposite classroom. For the Continuum Crowds method a 120x120 grid is used.

Two opposing groups scenario

	Streams model		Continuum Crowds	
	$\bar{x}$	s	$\bar{x}$	s
Reached goal (n)	200,0		200,0	
Collisions (n)	**0,3**	**0,6**	19,4	10,3
Time to goal (t)	**51,8**	**6,8**	61,8	13,5
Kinetic energy (mv^2)	**83,1**	**9,5**	99,1	10,3
Degrees turned (deg)	1,41	0,64	1,58	0,36
Maximum angular speed (deg/t)	**3,26**	**1,76**	5,18	0,70
Angular speed change (n)	**0,91**	**0,99**	1,39	1,57
Distance traveled (s)	**686,6**	**82,0**	766,5	83,1
Speed change (v)	**2,2**	**0,8**	99,3	33,6
Maximum speed change (a)	**0,19**	**0,06**	1,49	0,64
Significant speed change (n)	**0,00**	**0,00**	0,26	0,68
Average speed (v)	**1,33**	**0,04**	1,24	0,36

Table 3: Two groups of 100 agents moving towards the start location of the opposite group. For the Continuum Crowds method a 60x60 grid is used.

Two squeezing groups scenario

	Streams model		Continuum Crowds	
	$\bar{x}$	s	$\bar{x}$	s
Reached goal (n)	198,0		79,0	
Collisions (n)	6,2	5,5	5,1	6,1
Time to goal (t)	**74,7**	**17,2**	194,1	109,3
Kinetic energy (mv^2)	**76,4**	**9,8**	155,3	52,7
Degrees turned (deg)	**1,21**	**0,61**	1,55	0,24
Maximum angular speed (deg/t)	**2,34**	**0,34**	4,84	1,06
Angular speed change (n)	0,45	0,55	0,35	0,53
Distance traveled (s)	**757,0**	**143,1**	1579,1	673,3
Speed change (v)	**8,1**	**4,0**	494,1	368,0
Maximum speed change (a)	**0,24**	**0,06**	1,25	0,43
Significant speed change (n)	0,00	0,00	0,06	0,48
Average speed (v)	**1,02**	**0,12**	0,81	0,24

Table 4: Two groups of 100 agents squeezing through a narrow alley. For the Continuum Crowds method a 120x120 grid is used.

Two perpendicular groups scenario

	Streams model		Continuum Crowds	
	$\bar{x}$	s	$\bar{x}$	s
Reached goal (n)	400,0		387,0	
Collisions (n)	**1,1**	**1,7**	6,9	5,9
Time to goal (t)	**37,5**	**13,2**	69,0	25,8
Kinetic energy (mv^2)	**43,2**	**13,6**	89,7	13,6
Degrees turned (deg)	**1,33**	**0,35**	1,35	0,54
Maximum angular speed (deg/t)	**2,01**	**1,15**	5,42	0,23
Angular speed change (n)	0,84	0,58	0,92	1,32
Distance traveled (s)	**414,3**	**131,0**	757,0	126,4
Speed change (v)	**3,2**	**2,0**	152,8	65,6
Maximum speed change (a)	**0,18**	**0,09**	1,37	0,33
Significant speed change (n)	**0,00**	**0,00**	0,03	0,17
Average speed (v)	**1,104**	**0,129**	1,097	0,282

Table 5: Two groups of 200 agents crossing each other perpendicular. For the Continuum Crowds method a 60x60 grid is used.

Four crossing groups scenario

	Streams model		Continuum Crowds	
	$\bar{x}$	s	$\bar{x}$	s
Reached goal (n)	740,0		385,0	
Collisions (n)	**11,2**	**7,8**	18,2	11,2
Time to goal (t)	102,7	15,6	**89,2**	**16,8**
Kinetic energy (mv^2)	**113,3**	**14,9**	144,1	13,6
Degrees turned (deg)	0,92	0,54	0,92	0,60
Maximum angular speed (deg/t)	**2,99**	**1,64**	4,55	1,52
Angular speed change (n)	0,51	0,58	0,43	0,61
Distance traveled (s)	**1074,8**	**141,7**	1150,0	131,0
Speed change (v)	**10,8**	**3,7**	118,7	58,6
Maximum speed change (a)	**0,24**	**0,08**	1,27	0,53
Significant speed change (n)	**0,00**	**0,00**	0,10	0,33
Average speed (v)	1,047	0,079	**1,289**	**0,078**

Table 6: Four groups of 100 agents, each on one corner with the opposite corner as a goal position. For the Continuum Crowds method a 60x60 grid is used.

One group in a military environment scenario

	Streams Model	Continuum Crowds		
		60x60	120x120	240x240
	$\bar{x}$	$\bar{x}$	$\bar{x}$	$\bar{x}$
Reached goal (n)	200,0	200,0	197,0	159,0
Collisions (n)	**0,8**	1,1	1,5	3,6
Time to goal (t)	125,3	138,7	140,0	118,3
Kinetic energy (mv^2)	206,7	229,7	212,1	**193,3**
Degrees turned (deg)	0,8	0,8	0,8	1,2
Maximum angular speed (deg/t)	**1,3**	3,6	3,7	3,8
Angular speed change (n)	0,3	0,3	0,3	0,4
Distance traveled (s)	**1687,1**	1846,3	1774,9	1707,4
Speed change (v)	**3,5**	125,7	136,4	146,2
Maximum speed change (a)	**0,3**	0,8	0,9	0,9
Significant speed change (n)	0,0	0,0	0,0	0,0
Average speed (v)	1,3	1,3	1,3	**1,4**

Table 7: One group of 200 agents moving from to to bottom through an military environment. For the Continuum Crowds method three different grid sizes are used. In this case a t-test was done to compare each of the grid sizes of the Continuum Crowds with the Streams model.

In Table 8 we can see that the calculation time for a single frame is quiet similar for both methods. These results should however be taken with a grain of salt, since the calculation time of the Continuum Crowds model is dominated by the grid size.

	Streams Model	Continuum Crowds
Eight classrooms	231 ms	189 ms
Two opposing groups	84 ms	77 ms
Two squeezing groups	134 ms	137 ms
Two perpendicular groups	34 ms	23 ms
Four crossing groups	140 ms	190 ms

Table 8: Some results of the calculation times of both methods. The results are in milliseconds per frame.

To show that it is actually true that the Continuum Crowds is greatly dependent on the grid size, we conducted some experiments using the classroom scenario with different grid sizes and number of agents. As can be observed in Figure 7 the *dynamic potential field* is the most expensive step (up to 90% of the total frame calculation time) and scales quadratic with the grid dimensions.

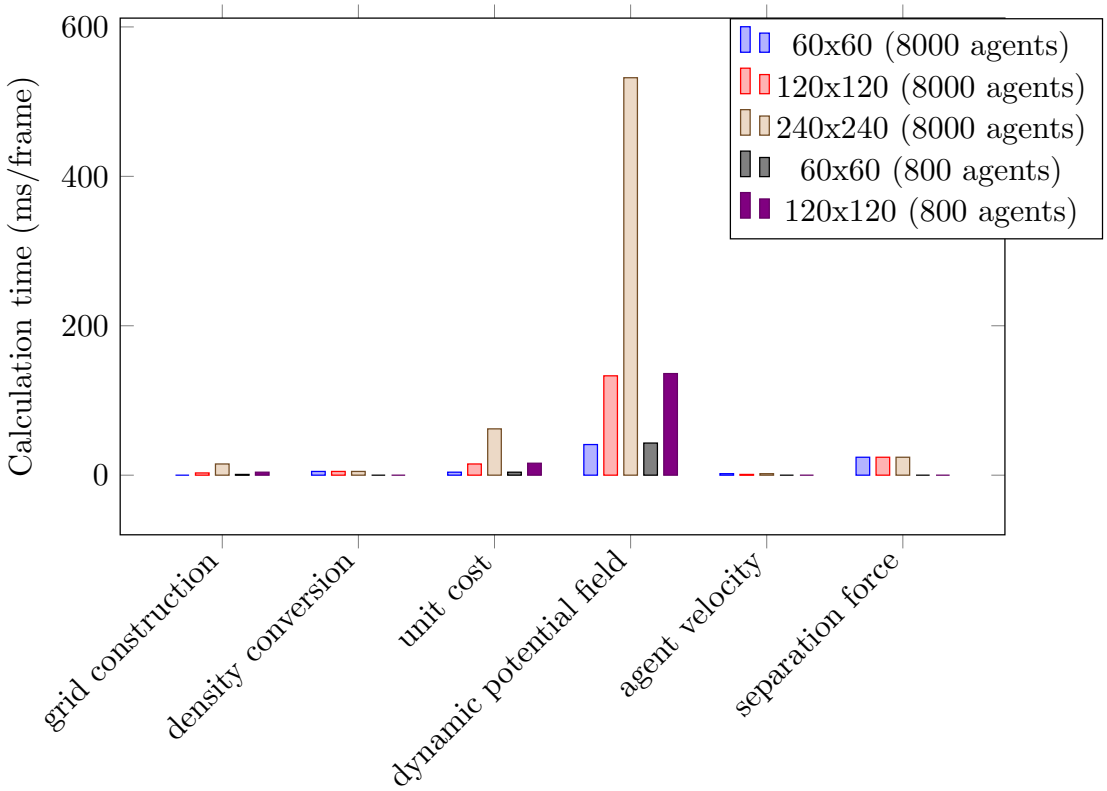

Figure 7: Calculation times for each step of the Continuum Crowds model.

6 Conclusion

In this paper we tried to answer the question what the advantages of the Continuum Crowds model are compared to the Stream model, and vice versa. We conducted several experiments to be able to define these advantages and disadvantages. To answer the main research question the fundamental limitations of both methods must be brought to light. The Continuum Crowds model has several limitations compared to the Streams model. The main limitations of the Continuum Crowds model are the use of a *grid* and the *pair-wise separation force*. The use of a grid limits the representation of the environment and the calculation time in all scenarios to the grid dimensions. Another limitation because of the use of a grid, is the suggested separation force of the authors, because this way collision avoidance can only be done up to the resolution of the grid. Furthermore this separation force can give unwanted results, like agents pushing each other with an invisible force. The other limitations of the Continuum Crowds model are the assumptions that agents can perceive global information and that agents move as a group to a common goal. Additionally, in real crowds people are not equipped with information (like a grid with unit costs) other than their own perceived information. Also the assumption of a common goal could be not manageable if people stroll without a group specific goal.

The Streams model can use any global planning method and any local collision-avoidance method, which makes the model greatly depend on the chosen methods. This way there could be conflicting requirements between the local collision avoidance and the global planner, but this is a problem that is beyond the scope of the Streams model. The Streams model furthermore limits the perception of global information to the field of view of the agent and does not handle visual occlusions by other agents. The assumption of the field of view for each agent probably gives less information about the environment than in reality. Real people for example also use sound to determine surrounding people and people can turn their head to perceive the surrounding environment. This last problem could be resolved by uncoupling the field of view from the direction of the velocity or by enlarging the angle of the field of view.

The advantages of the Continuum Crowds model over the Streams model is the unification of the global path planner and the local collision avoidance. This means that there cannot be conflicting requirements between those. The second advantage over the Streams model is the use of global knowledge instead of only the field of view of an agents. In large crowds this could mean that the Continuum Crowds model is more smooth and has fewer deadlocks because of this global information, unless there are crossing flows as can be seen in Table 3-6.

An advantage of the Streams model over the Continuum Crowds model is the flexibility and individual variability of agents. In the Streams model agents can have different parameters, while the Continuum Crowds model assumes homogeneous agents. When analyzing the observations in the results section, the Streams model gives more smooth paths and has less collisions. This could be due to the fact that the Continuum Crowds model uses a uniform grid and a pair-wise separation force.

Based on the results of the performed experiments we can see that the Continuum Crowds model has low computational costs if the number of groups is small, but becomes less profitable when a large number of groups with different goals is used. On the other hand, groups with a large number of agents can be easily handled by the Continuum Crowds model, while many individuals with different goals can be handled better with the Streams model. The calculation time per frame during the experiments (which were only conducted using up to eight different groups) are fairly similar, but again mainly depend on the grid size for the Continuum Crowds model. On average the Streams model results in more smooth paths, but this is again also greatly depending on the grid size of the Continuum Crowds model.

Both methods could be useful in different scenarios. The Continuum Crowds model is probably more advantageous in situations with huge numbers of agents which all have one common goal, while the Streams model handles large numbers of groups with all different goals. For complex environments the Streams model would be more advantageous since it

is not dependent on a global planner with a grid, but in simple environments with a low number of obstacles the Continuum Crowds model is probably faster. The choice of the model thus is entirely dependent on the application for which it is used.

6.1 Combining both methods

A combination of both methods could also be beneficial in some situations. The bottleneck of the Continuum Crowds model is the calculation of the dynamic potential field every defined time-step. Using the field of view (as explained in the Streams method) we could calculate the density and velocity, making the computation of the new 'dynamic potential field' only required when the goal of the group changes. Since the conversion to a density and velocity field is not needed anymore, the unit cost would be equal in each cell. We now can use the Fast Marching Algorithm [18] as described in Section 3.1 to compute the new potential field. This potential field can now be used as input for the Streams model, since we can calculate an individual velocity for each agent. This could lead to better performance and agents that use the follow and alignment strategy as described by Van Goethem [7].

A global planner for large groups without using a computational expensive grid is also imaginable. The idea of the Fast Marching Algorithm is also useful for global planners with an underlying graph or navigation mesh. The potential field could similarly be calculated, propagating from the goal location, covering the entire free space. This way the grid becomes superfluous. This dynamic field, to calculate the individual velocities for all agents, could now be used as input for the Streams method.

References

[1] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[2] Craig W Reynolds. Steering behaviors for autonomous characters. In *Game developers conference*, volume 1999, pages 763–782, 1999.

[3] Mehdi Moussaïd, Dirk Helbing, and Guy Theraulaz. How simple rules determine pedestrian behavior and crowd disasters. *Proceedings of the National Academy of Sciences*, 108(17):6884–6888, 2011.

[4] Samuel Lemercier, Asja Jelic, Richard Kulpa, Jiale Hua, Jérôme Fehrenbach, Pierre Degond, Cécile Appert-Rolland, Stéphane Donikian, and Julien Pettré. Realistic following behaviors for crowd simulation. In *Computer Graphics Forum*, volume 31, pages 489–498. Wiley Online Library, 2012.

[5] Sachin Patil, Jur Van Den Berg, Sean Curtis, Ming C Lin, and Dinesh Manocha. Directing crowd simulations using navigation fields. *Visualization and Computer Graphics, IEEE Transactions on*, 17(2):244–254, 2011.

[6] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 1160–1168. ACM, 2006.

[7] Arthur van Goethem and Roland Geraerts. A stream algorithm for crowd simulation to improve crowd coordination at all densities. 2012.

[8] Shawn Singh, Mubbasir Kapadia, Petros Faloutsos, and Glenn Reinman. Steerbench: a benchmark suite for evaluating steering behaviors. *Computer Animation and Virtual Worlds*, 20(5-6):533–548, 2009.

[9] Dirk Helbing, Illés J Farkas, and Tamás Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, 2000.

[10] Dirk Helbing, Illés J Farkas, and Tamás Vicsek. Freezing by heating in a driven mesoscopic system. *Physical review letters*, 84(6):1240, 2000.

[11] Nuria Pelechano, Jan M Allbeck, and Norman I Badler. Controlling individual agents in high-density crowd simulation. *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 99–108, 2007.

[12] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1928–1935, 2008.

[13] Avneesh Sud, Erik Andersen, Sean Curtis, Ming Lin, and Dinesh Manocha. Real-time path planning for virtual agents in dynamic environments. page 55. ACM, 2008.

[14] Roland Geraerts. Planning short paths with clearance using explicit corridors. pages 1997–2004, 2010.

[15] Craig Reynolds. Big fast crowds on ps3. *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*, pages 113–121, 2006.

[16] Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C Lin. Aggregate dynamics for dense crowd simulation. *ACM Transactions on Graphics*, 28(5):122, 2009.

[17] Dave Ferguson and Anthony Stentz. Field d*: An interpolation-based path planner and replanner. pages 239–253, 2007.

[18] John N Tsitsiklis. Efficient algorithms for globally optimal trajectories. *Automatic Control, IEEE Transactions on*, 40(9):1528–1538, 1995.

[19] Armin Seyfried, Bernhard Steffen, Wolfram Klingsch, and Maik Boltes. The fundamental diagram of pedestrian movement revisited. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(10), 2005.

[20] Stephen J Guy, Jatin Chhugani, Sean Curtis, Pradeep Dubey, Ming Lin, and Dinesh Manocha. Pledestrians: a least-effort approach to crowd simulation. *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 119–128, 2010.

[21] Jan Ondřej, Julien Pettré, Anne-Hélène Olivier, and Stéphane Donikian. A synthetic-vision based steering approach for crowd simulation. *ACM Transactions on Graphics*, 29(4):123, 2010.

[22] Vassilios Kountouriotis, Stelios CA Thomopoulos, and Yiannis Papelis. An agent-based crowd behaviour model for real time crowd behaviour simulation. *Pattern Recognition Letters*, 44:30–38, 2014.

[23] Ioannis Karamouzas, Roland Geraerts, and Mark Overmars. Indicative routes for path planning and crowd simulation. *Proceedings of the 4th International Conference on Foundations of Digital Games*, pages 113–120, 2009.