Density Constraints for Crowd Simulation

Ioannis Karamouzas, Jiri Bakker and Mark H. Overmars
Department of Information and Computing Sciences, Utrecht University
Padualaan 14, De Uithof, 3584 CH Utrecht, The Netherlands
{*ioannis, jgbakker, markov*}*@cs.uu.nl*

Abstract

Virtual worlds are nowadays commonly used in interactive applications, like computer games and simulations. Typically, such worlds are populated by a large number of virtual characters. On one hand, these characters have global goals with respect to the environment and thus, they must be able to plan their paths toward their desired locations. On the other hand, they should try to avoid collisions with each other and with the environment. In this paper, we present a new technique to improve the global routes that characters prefer to follow by taking into account the crowd density. Consequently, in our simulations, characters can intelligently plan their paths around congested areas, favoring less dense regions. This leads to an efficient flow of characters, reducing at the same time the amount of energy and time consuming avoidance maneuvers that the characters have to perform. The technique is fast and can plan paths for thousands of characters in real-time.

1. Introduction

Virtual worlds are nowadays commonly used in computer games, training applications and on-line communities. Such worlds are inhabited by a large number of moving characters. These characters, on one hand, must plan their paths between different locations in the world. On the other hand, they should be able to move toward their desired locations in a human-like manner, avoiding collisions (and near collisions) with each other and with the environment. As a result, visually convincing and physically correct simulations of large crowds have become a necessity for interactive virtual worlds and games.

In this paper, we improve the global routes that characters prefer to follow by taking the crowd density into account. This not only leads to a smoother and more effective character flow, but also reduces the amount of time and energy consuming interactions between characters.

1.1. Related Work

Path planning has been extensively studied over the past years and many models have been proposed to simulate individuals, groups and crowds of characters.

In the game development community the most common approach is to divide the environment into a grid of cells and search for a free path using A* based algorithms [1], [2]. This approach guarantees that a path, if one exists, will always be found. However, the paths returned by A* algorithms tend to be aesthetically unpleasant, requiring a lot of time-consuming smoothing. In addition, grid-based searches can become computationally expensive, especially for large and complicated environments.

An alternative approach is to use a navigation mesh [1], [3] for the global navigation of the characters. Navigation meshes partition the terrain into convex polygons that represent the walkable area of the environment. The static part of the game world is taken into account upon the construction of the mesh, and thus, the character only has to consider collisions with dynamic obstacles. Similar to a grid-map, a navigation mesh consists of linked cells sharing common edges. Therefore, the character's path can be retrieved by applying an A* search algorithm, leading to the same drawbacks as the ones discussed above.

The path planning problem becomes even harder when many characters populate the virtual world, since each character has to move toward its goal position avoiding at the same time other characters and dynamic obstacles. For that reason, collision avoidance is typically achieved using a reactive steering approach (see for example [4], [5]).

In reactive steering, the character adapts its previously computed motion to the obstacles found along its

(a) The British cavalry should move toward the area indicated by the green line.

(b) A number of horsemen get stuck while trying to pass through a narrow passage.

Figure 1: One of the main problems with reactive-based planners is that the characters can get stuck in cluttered environments and never reach their targets. This example was taken from the game Empire: Total War published by Sega.

path. This method can handle large dynamic environments providing enough flexibility for the characters to avoid local hazards. However, the local nature of the method gives no guarantees on the resulting behavior. The characters run the risk of getting stuck in local minima and not being able to reach their goals (see Figure 1 for an example). This normally leads to deadlock situations that can only be resolved by rather unnatural motions, as can be observed in many modern RTS games like the latest Empire: Total War title [6], or in recent action-adventure games like the Grand Theft Auto IV [7].

Reactive navigation techniques originate from the robotics community and are based on variants of potential-field approaches [8], [9], [10]. In the gaming and animation community, the concept of reactive planning was introduced by the seminal work of Reynolds on *boids*, often referred to as flocking [11]. Reynolds used simple local rules to describe the collective behavior of bird flocks and fish schools. Later, he extended the technique to include additional steering behaviors [5]. His approach works very well in open environments and generates interesting movements. However, the characters can easily get stuck behind obstacles in cluttered environments, since only local information is taken into account.

Modeling of emergent collective phenomena has also received a lot of attention in civil and traffic engineering. The social force model proposed by Helbing and Molnár [12] has been influential in this field. Helbing used social forces to simulate the behavior of pedestrians and traffic flow. Since his original work, many interesting models have been proposed that exhibit emergent crowd behaviors (e.g. [13]). Nevertheless, all these approaches are closely related to flocking and reactive steering and suffer from the same local-minima problems.

To address this, several local models have been combined with global navigation techniques, such as the methods based on Corridor Maps [14] and Adaptive Elastic Roadmaps [15]. Although these approaches are fast and flexible, they rather limit the global behavior of the character, since a graph-based method is used for global planning.

An alternative approach has been proposed by Treuille *et al.* [16]. Their model unifies global navigation and local collision avoidance into a single framework and is able to reproduce specific crowd phenomena. However, it is limited to homogeneous groups of characters moving toward a common goal. In contrast, we focus on independent characters that have distinct characteristics and goals.

More recently, Karamouzas *et al.* have proposed the Indicative Route Method (IRM) as a new approach for path planning and crowd simulation [17]. In the IRM a so-called *indicative route* determines the preferred (global) route of the character. Such a route can be indicated manually by the level designer or computed automatically using, for example, an A* algorithm on a coarse grid. Then, a force-field approach guides the character through an obstacle-free area (corridor) around this route, leading to a smooth path. The IRM has been successfully used to steer in real-time thousands of characters through complex virtual worlds. In

addition, by exploiting the notion of indicative routes, the method provides the required flexibility in the desired routes of the characters.

Another interesting model for pedestrian simulation has been proposed by Loscos *et al.* [18] that is based on simple grid-based rules. In this approach, a 2D grid subdivision is used for collision detection and avoidance. To increase the realism of the local interactions, the directions of the characters are also stored into a 2D grid structure and are dynamically updated during the simulation. These directions are then used to provide coordination among the characters, allowing the emergence of pedestrian flows.

Similar to this idea, Chenney [19] used divergence-free flow tiles to guide the motions of the characters through a virtual world without requiring any form of collision detection. Due to the static nature of the tiles, though, no interactions among the characters can take place.

More recently, Jansen and Sturtevant proposed a new approach to cooperative pathfinding by adding an extra penalty to the A* cost function when characters try to move against the flow [20]. Consequently, the characters are encouraged to follow the directions that are taken by previous characters, which results in emergent cooperative behavior. However, as the number of the characters grows, the problem complexity and the running time increases considerably, making this approach insufficient to model large crowds. Furthermore, as mentioned above, the paths resulting from A* algorithms can be of low quality and thus, extra care should be taken to smooth them.

1.2. Contributions

Our model is inspired by the work of Jansen and Sturtevant [20] and our previous work on indicative routes [17]. We focus, though, on the crowd density rather than on the flow field. Our goal is not to simulate how real pedestrians behave in a crowd, but to provide a crowd model that looks pleasing and convincing to the viewer. Therefore, in our simulations, we aim for characters that prefer to move toward low-density areas. This leads to energy-efficient paths and a better spread of the characters throughout the environment. Our approach allows the users to set the trade-off between path length and preferred density, making it a flexible framework that can simulate a wide variety of character behaviors, even for characters in the same scene.

Globally speaking, our method creates a *density map* that provides information regarding the density of the environment. This density map consists of a coarse grid in which each cell stores information about the number of characters that traversed this cell in the recent past. Using an A* search on the density map, a path for the character is computed. As indicated above, though, such a path can be rather unrealistic and might not avoid other moving characters. Hence, we do not use this path directly, but as an indicative route. We then apply the Indicative Route Method to guide the character toward its goal position. As a result, a high-quality and collision-free path is generated that avoids high-density areas, keeping at the same time the computational cost of our approach low, since we only need a coarse density map.

1.3. Outline of the paper

The rest of the paper is organized as follows. Section 2 explains our proposed approach for simulating crowds of virtual characters. Experiments to show the usability of our method are discussed in Section 3. Finally, some conclusions and plans for further research are presented in Section 4.

2. Density Adaptation

In this section we provide a detailed description of our proposed approach. It is based on a simple idea that characters adapt their path planning choices depending on the density of the environment. This leads to characters that can intelligently plan their routes, avoiding areas of high congestion and favoring less crowded regions.

2.1. Density Maps

To incorporate the notion of crowd density into our model, a density map is constructed. This map discretizes the world into a regular grid and for each grid cell a scalar value is assigned indicating the density of this cell. We refer the reader to Figure 2 for an example of a density map.

The density map is updated in real-time based on the actual motions of the characters. In the beginning of the simulation all cells have a zero density value. As soon as a character steps into a grid cell or moves within the cell, its density is updated and a new value ρ_N is stored by taking the current density value ρ_C into account, that is:

$$\rho_N = \rho_C + \rho_I, \tag{1}$$

where the parameter ρ_I controls how fast the density value of the cell increases.

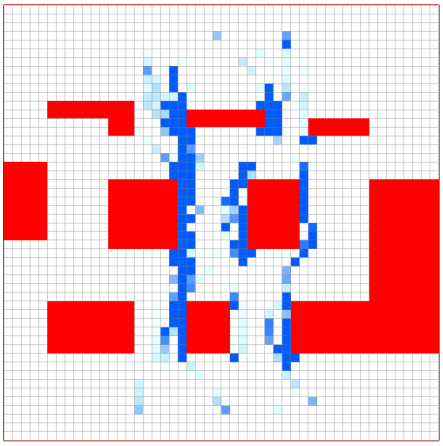


Figure 2: An example of a density map used in our simulations. Darker cells represent areas with high crowd density.

At any time step t of the simulation, the (current) density $\rho(t)$ of a grid cell can be computed as follows:

$$\rho(t) = \rho_o - \Delta t * \rho_D, \quad (2)$$

where ρ_o is the previously stored density value and Δt is the amount of time in seconds passed since the last update. The parameter ρ_D controls the speed of the density fall off. This means that the density field degrades over time and eventually disappears, allowing for a character to take into account only cells that are recently traversed by other characters (note that the density $\rho(t)$ is clamped to non negative values). The method is very efficient, as computations are only performed when a character visits a cell.

2.2. Computing a Global Route

Given the start and the goal position of the character, its route can now be computed by applying an A* based search on the free grid cells. In general, the A* algorithm repeatedly examines the most promising locations on the grid to find the lowest cost path. Therefore, a cost function $f(N)$ is associated with each cell N that characterizes the cost of the path up to this cell. The cost function is defined as

$$f(N) = g(N) + h(N), \quad (3)$$

where $g(N)$ determines the cost from the start to the current cell, and $h(N)$ is the heuristic function that estimates the minimum path cost between the current location and the goal.

Typically, an A* search algorithm minimizes the distance that the character has to travel and returns the shortest path between the start and the goal position. However, since we strive for paths that avoid crowded areas, the density should also be taken into account upon planning. Therefore, the density map is used to modify the g-cost of a cell as follows

$$g(N) = g(N-1) + d(N, N-1) + \lambda \rho_N, \quad (4)$$

where $d(N, N-1)$ is the distance between the current cell and its predecessor, ρ_N is the density value of the current cell provided in (2) and λ is a weighting parameter controlling the balance between density and path length.

Since we cannot predict the density in the remaining part of the path, the heuristic function only estimates the remaining distance to the goal, that is

$$h(N) = d(N, goal) \quad (5)$$

As this is a lower bound to the actual cost and our cost function is always positive, the heuristic is always admissible. Hence, the method is guaranteed to find the optimal path.

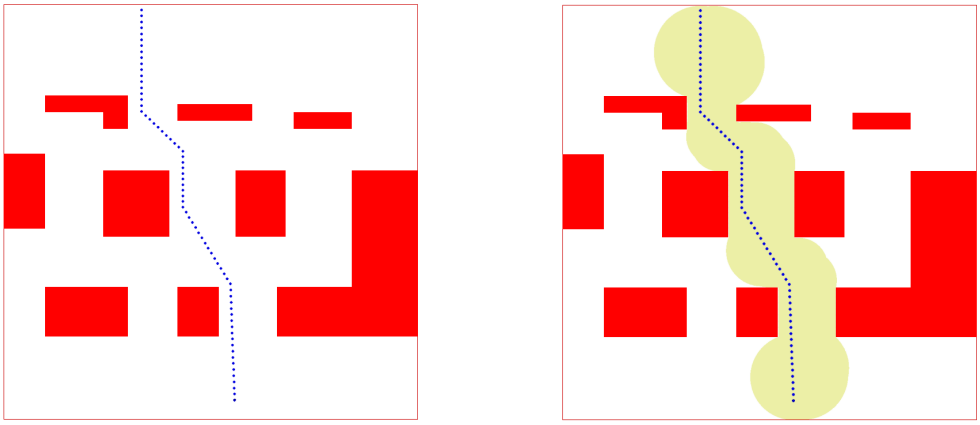
2.3. Local Navigation and Final Motion

Having retrieved the global route of the character, we can now plan its final motion using the Indicative Route Method (IRM) as proposed in [17]. One could argue that the character can directly follow the path returned by our modified A* algorithm. However, the resulting motion may not be collision-free. In addition, A* based paths are, in general, not natural. Natural paths typically follow smooth curves, keep a preferred amount of clearance from obstacles and are flexible enough to avoid other characters and moving entities.

The IRM satisfies all these criteria. It can generate high-quality and believable paths. In addition, it is fast and flexible, allowing the simulation of thousand of characters in real-time. We refer the reader to [17] for a detailed explanation of the IRM.

We now briefly explain how our proposed approach can be integrated into the IRM. We begin by querying the grid map in order to retrieve a path, as discussed in the previous section. This path defines the so-called *indicative route* of the character and provides an indication/rough estimation of the character's preferred route. An example of an indicative route is shown in Figure 3(a).

The character does not traverse exactly its indicative route, but rather use it as a guide to safely navigate toward its desired location. For that reason, a *corridor* is constructed around this route, indicating the area in which the character can move without colliding with the environment (see Figure 3(b) and [17] for



(a) Indicative route (b) Corridor

Figure 3: An indicative route and its corresponding corridor.

more details). Once the corridor is created, a force-field approach drives the character's motion through the corridor. In particular, several social forces are acted upon the character and influence its movements using basic physics laws.

First, a *steering force* advances the character toward its goal. To this end, an attraction point moves along the indicative route and attracts the character. Hence, the character is steered to less dense areas. Second, a *boundary force* is applied that pushes the character away from the boundary of the corridor. This ensures that the character remains inside its corridor. To avoid collisions with other characters and moving entities, an additional *collision avoidance force* is required. Therefore, in our simulations we have used part of the social force model proposed by Helbing and Molnár in [12] which is known to exhibit emergent and realistic crowd behavior.

Given the applied forces on the character, time integration is used to update the character's velocity and position, resulting in a smooth path that stays inside the corridor, moves toward the goal, and avoids other characters.

3. Experiments and Discussion

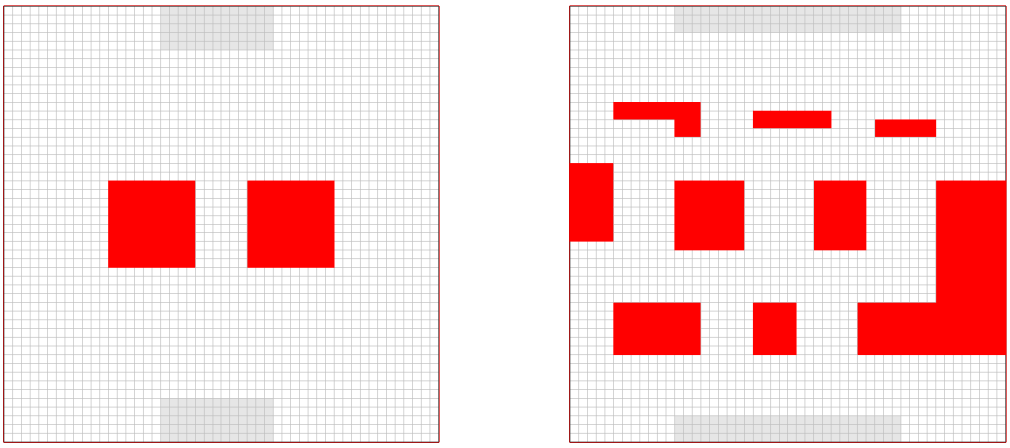
We have implemented our proposed approach to experimentally test its effectiveness and applicability in interactive applications, like computer games and simulations. All the experiments were performed on a PC running Windows XP, with a 2.4 GHz Intel Core2 Duo CPU and 2 GB memory.

The experiments were conducted for the two environments depicted in Figure 4. The first environment is very simple. Characters would typically move through the corridor in the center, which will eventually become cluttered. We expect that our new approach will steer the characters around the obstacles. In the second environment there are many more possible routes that a character can follow to reach its destination. We expect that our method will allow for variations in the characters' routes as congestion evolves across the scene.

For both environments a 50 x 50 grid map was used, with each cell covering an area of 2 x 2 meters. In our simulations, characters were represented as discs having radius of $0.6\ m$ and a maximum speed of $1.4\ m/s$. This corresponds to the average walking speed of humans [21]. To obtain visually pleasing simulations, the density increase parameter in (1) was set to 0.2 ($\rho_I = 0.2$) and the time decay factor in (2) to 0.1 ($\rho_D = 0.1$).

To determine the quality of the generated paths, we selected a varying number of characters and placed them randomly at the top and the bottom of the two environments. Each character had to advance toward a random goal position on the other side of the environment (see Figure 4). New characters were spawn every 3 seconds. Simulations were run for a maximum number of 30,000 iterations and every iteration calculated 0.1 seconds of characters' movements.

Figure 5(a) shows an example of a density map obtained using an A* shortest path algorithm (note that the IRM was used to plan the final motion of the character). As can be inferred from the figure, all characters take the shortest path and move through the narrow corridor between the two obstacles. As the environment becomes more crowded, this leads to inefficient movements and eventually characters get



(a) (b)

Figure 4: The two test environments. Gray areas indicate the start and goal positions of the characters.

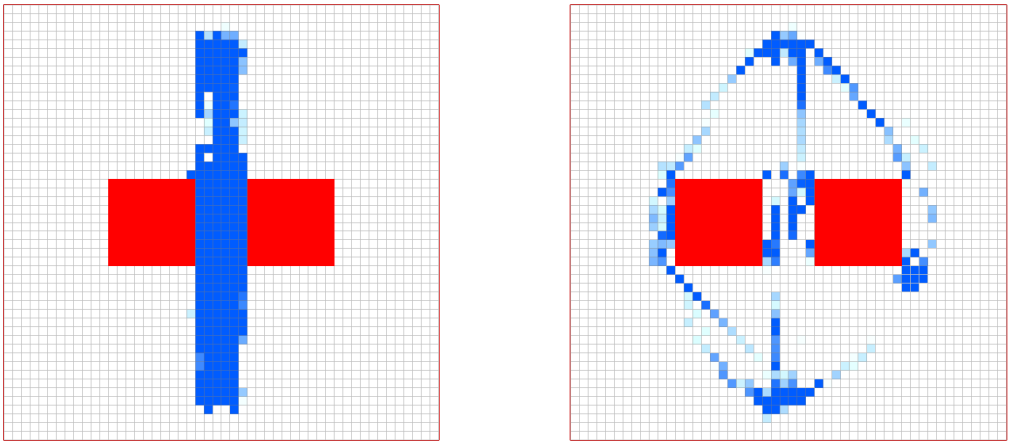
stuck in the congested area. Figure 5(b) shows an example of a density map obtained using our approach. Clearly, the characters anticipate that a congestion will occur and try to avoid it in advance by moving around the obstacles instead of moving between them, which leads to an effective character flow. Examples of density maps for the other test environment are depicted in Figure 5. Again, it can be observed that the characters spread further over the environment.

To quantitatively compare the two approaches a number of metrics have been devised. In particular, we computed the total time that a character needs to reach its goal, as well as the average speed with which the character moves. We are also interested in how energy-efficient are the movements of a character. For that reason, we used the total acceleration as a measure of the movement effort spent by a character [22]. The average acceleration was also reported as a measure of time-independent effort.

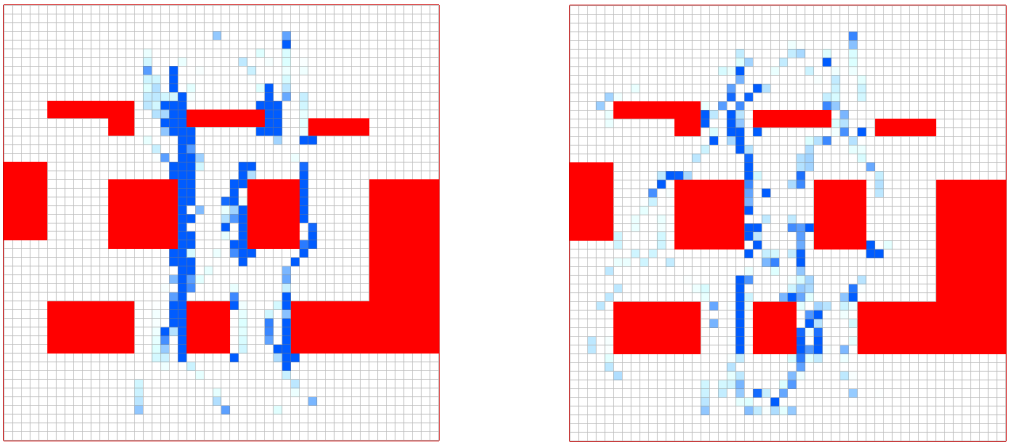
Table 1(a) summarizes the results we obtained for the simple environment shown in Figure 2(a). Note that for our approach, the weighting parameter λ in the A* cost function was set to 3 ($\lambda = 3$). The numbers reported in the table are the averages over 10 simulation runs. As indicated above, each simulation consists of $30,000$ iterations. However, to ensure that the density map has reached into a stable state, we first let each simulation run for $10,000$ iterations and then start gathering the detailed statistics of the characters. A statistical analysis (t-test using a significance level of 5%) was performed afterwards to determine how significant were the results.

The analysis has shown that the time required to reach a destination is higher using our approach, ($M = 79.654, SD = 10.555$) than by taken only the path length into account ($M = 71.567, SD = 8.706$), $p < 0.001$. This is expected, since the characters take some detours to avoid crowded areas. Consequently, they move with a higher speed ($M = 1.31, SD = 0.111$) than characters that follow shortest paths ($M = 1.258, SD = 0.094$), $p < 0.001$. In addition, using our approach, the number of avoidance maneuvers that the characters have to perform is reduced. This leads to more energy-efficient paths, as the characters have to spend less effort to avoid potential collisions. In particular, the total acceleration is significantly lower ($M = 699.965, SD = 261.131$) than using the A* shortest algorithm ($M = 770.939, SD = 262.198$), $p < 0.001$. The same also applies to the average acceleration of the characters.

Similar conclusions can also be drawn for the complex environment, as can be inferred from Table 1(b). Note that this environment offers more variation in the routes that the characters can follow. As a result, using the density map approach, the characters plan early for congestion and select paths that move through less crowded locations. This reduces the number of interactions between the characters, leading also to higher walking speeds (the differences in average acceleration, total acceleration and speed are statistically significant, i.e. $p < 0.001$). Consequently, although the characters spread out across the environment and take longer paths, they need less time to travel toward their goals ($M = 78.55, SD = 12.803$) than characters that prefer shortest paths ($M = 81.439, SD = 15.693$), $p = 0.01$. This is not the case in the simple environment, since each character has only three global route choices. It can either move through the narrow



(a) Shortest Path (b) Density



(c) Shortest Path (d) Density

Figure 5: Example of density maps obtained by combining an A* shortest-path search algorithm or our modified A* algorithm that takes the crowd density into account with the Indicative Route Method.

corridor or select the path that is at the right or left of the corridor. Thus, due to the lack of alternative routes, the environment becomes easily crowded and hence, characters that take the shortest path need less time than characters that also take density into account.

Besides the quantitative analysis, we have also empirically compared the two approaches by observing the resulting simulations. We encourage the reader to view the related video that is available at *http://people.cs.uu.nl/ioannis/crowd-density*. As we used part of the Helbing's social force model for the local interactions of the characters, we observed in both methods the phenomenon of lane formation that is widely noted in pedestrian literature [23]. However, using our approach, the flow of the characters looks smooth and visually pleasing and the paths of the characters are considerably less curved compared to the paths obtained using the IRM combined with an A* shortest path algorithm.

In conclusion, our approach, indeed, leads to a better spread of characters over the environment, less character interactions and more energy-efficient paths, resulting in a more pleasing simulation.

4. Conclusions

In this paper we have presented a new approach toward a more realistic simulation of crowds of characters. In our model, characters adapt their planning behavior depending on the crowd density. For that reason, the notion of density map was introduced as a data structure to store density information. Given a density map, a global route for the character is retrieved by applying an A* search algorithm that takes

	Time		Avg.Speed		Total Acceleration		Avg. Acceleration	
	Mean	Sdev	Mean	Sdev	Mean	Sdev	Mean	Sdev
Shortest Path + IRM	71.567	8.70	1.258	0.094	770.939	262.198	1.329	0.315
Density + IRM	79.654	10.555	1.310	0.111	699.965	261.131	1.085	0.328

(a) Simple Environment

	Time		Avg.Speed		Total Acceleration		Avg. Acceleration	
	Mean	Sdev	Mean	Sdev	Mean	Sdev	Mean	Sdev
Shortest Path + IRM	81.439	15.693	1.206	0.147	910.432	383.845	1.358	0.351
Density + IRM	78.55	12.803	1.307	0.104	694.65	246.01	1.096	0.286

(b) Complex Environment

Table 1: Time, speed and acceleration statistics using the shortest path as the indicative route of the character and a path that takes into account both density and path length. The reported numbers are the averages over 10 simulations, each involving 325 characters.

into account both density and path length. The final motion of the character is then obtained using the Indicative Route Method. This leads to high-quality and collision-free paths, ensuring at the same time visually convincing motions.

We have experimentally shown that, using our approach, characters can intelligently plan around crowded areas, preferring to move through less dense regions. Although the resulting paths are longer, the amount of local interactions between characters is reduced, which leads to time and energy efficient movements.

During our experiments, an initial phase was used to ensure that the density map has converged to a stable state. In real-time applications like games, the initial state of the map can be computed in a preprocessing phase or provided manually by the level designer to encourage certain character behavior. While the player-controlled character moves through the virtual world and interacts with other characters, the density map is dynamically updated as discussed in Section 2.1.

We are confident that interactive applications like computer games and virtual environment applications can benefit from our proposed approach. Currently, we are investigating how we can incorporate the notion of influence regions into our path planner. Such regions can either be dangerous places the characters prefer to avoid or appealing locations they would like to visit. We also simulate characters that prefer to move with the flow, by taking into account the direction in which previous characters have traveled, as proposed in [18], [20].

In all these cases a similar approach can be applied that places a multi-layer coarse grid over the environment, with each layer storing and maintaining information about different factors that influence a character's path, such as density, danger and/or flow. Since navigation meshes has gained a lot of popularity lately, an alternative approach is to store the influence values at the vertices of navigation mesh polygons. Next, an indicative route is determined using an A* search on the grid or on the mesh nodes, and, finally, the Indicative Route Method is applied.

Acknowledgment

This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie), and by the Metaverse1 project.

References

[1] M. DeLoura, *Game Programming Gems*. Charles River Media, Inc., 2000.

[2] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1994.

[3] S. Rabin, *AI Game Programming Wisdom 4*. Charles River Media, Inc., 2008.

[4] W. B. Stout, "Smart moves: Intelligent pathfinding," *Game Developer*, October 1996.

[5] C. W. Reynolds, "Steering behaviors for autonomous characters," in *Proc. of Game Developers Conference*, San Jose, California, 1999, pp. 763–782.

[6] *Empire: Total War*, Sega, 2009. [Online]. Available: http://www.totalwar.com/empire/

[7] *Grand Theft Auto IV*, Rockstar North, 2008. [Online]. Available: http://www.rockstarnorth.com/

[8] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, pp. 90–98, 1986.

[9] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential fields," *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 501–518, 1992.

[10] J. H. Reif and H. Wang, "Social potential fields: a distributed behavioral control for autonomous robots," in *WAFR: Proceedings of the workshop on Algorithmic foundations of robotics*, 1995, pp. 331–345.

[11] Reynolds, "Flocks, herds, and schools: A distributed behavioral model," *Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987, proceedings of SIGGRAPH '87.

[12] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, pp. 4282–4286, 1995.

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

[14] R. Geraerts and M. Overmars, "The corridor map method: A general framework for real-time high-quality path planning," *Computer Animation and Virtual Worlds*, vol. 18, pp. 107–119, 2007.

[15] A. Sud, R. Gayle, E. Andersen, S. Guy, M. Lin, and D. Manocha, "Real-time navigation of independent agents using adaptive roadmaps," in *ACM symposium on Virtual reality software and technology*, 2007, pp. 99–106.

[16] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 1160–1168, 2006.

[17] I. Karamouzas, R. Geraerts, and M. Overmars, "Indicative routes for path planning and crowd simulation," in *FDG '09: Proceedings of the 4th International Conference on Foundations of Digital Games*, 2009, pp. 113–120.

[18] C. Loscos, D. Marchal, and A. Meyer, "Intuitive crowd behaviour in dense urban environments using local laws," *Theory and Practice of Computer Graphics*, 2003.

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

[20] M. R. Jansen and N. R. Sturtevant, "Direction maps for cooperative pathfinding," in *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2008.

[21] R. Knoblauch, M. Pietrucha, and M. Nitzburg, "Field studies of pedestrian walking speed and start-up time," *Transportation Research Record*, vol. 1538, pp. 27–38, 1996.

[22] S. Singh, M. Naik, P. Faloutsos, and G. Reinman, "Steerbench: A benchmark suite for evaluating steering behaviors," in *Motion in Games, First International Workshop. Revised Papers*, vol. 5277. Springer-Verlag, 2008, pp. 200–209.

[23] D. Helbing and P. Molnár, "Self-organization phenomena in pedestrian crowds," in *Schweitzer, F. (Ed.), Self-Organization of Complex Structures: From Individual to Collective Dynamics*. Gordon and Breach, 1997, pp. 569–577.