Large Scale Crowd Simulation Using A Hybrid Agent Model

Seung In Park[1], Yong Cao[1], Francis Quek[1]

[1]Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, USA
{spark80,yongcao,quek}@vt.edu

Abstract. We present a hybrid model for large-scale crowd simulation by augmenting continuum dynamics under an agent based perspective. Our model supports both group behaviors and complex behaviors driven by individual decisions of agents. Our simulation system is implemented on the parallel architecture of graphics processing units (GPUs), and scales very well with respect to the size of the virtual environment. In our experiments, tens of thousands of characters can be simulated on a large grid, $4,096 \times 4,096$, at interactive rates.

1 Introduction

Realistic crowd simulation has become increasingly important in computer games, entertainment, architecture design, and numerous other applications. Real-time crowd simulation remains challenging because it should reflect essential features of crowd dynamics as well as support computational efficiency. A promising recent direction based on continuum dynamics was proposed by Hughes [7] and extended by Treuille et al. [18]. Continuum crowd model integrates global navigation with local collision avoidance as a single optimization problem, and achieves computational efficiency. In addition, a simulation created with the approach shows smooth flow of thousands of characters and produces emergent behaviors such as lane formation. However it sacrifices the individual variability in behaviors for reduced computational cost, and the simulations are limited to unrealistic cases where a group of homogeneous people share an identical goal and regular patterns of behavior. Furthermore, the computational efficiency of the continuum dynamics requires a compromise in the scale of the grid that can be simulated because of the complexity of solving the related Eikonal equations.

Agent-based methods are good alternatives to capture such a complex pedestrian interaction and movement with regard to individual characteristics. In order to provide more sophisticated control over agents, the agent-based systems include the reasoning process of path planning, steering and congestion planning separately for each agent. The drawback of such sophisticated simulation is that the computation cost is too high to support real-time simulation of a large crowd.

In this paper, we propose a hybrid model for large crowds by augmenting continuum dynamics under an agent based perspective, thus the crowd exhibits complex behavior driven by individual decisions of agents while the simulation achieves an efficient performance. The model is capable of driving tens of thousands of characters in large and high-resolution environments at interactive rates on current desktops. The specific contributions of this paper are: 1. We propose a hybrid crowd model to support individual behavior of agents and cost effective simulation by integrating agent-based modeling and continuum dynamics; 2. Our model supports a scalable crowd simulation of continuum model in large and high resolution environments at interactive rates; 3. Our model is highly parallel and amenable to GPGPU computing.

The remainder of this paper is organized as follows: Section 2 provides a brief overview of the related work in crowd simulation. We introduce our model of virtual environment in Section 3. Section 4 presents our hybrid approach for crowd navigation. We demonstrate and evaluate the approach both qualitatively and quantitatively in Section 5. We summarize the conclusions drawn from our studies, and provide possible future research directions in Section 6.

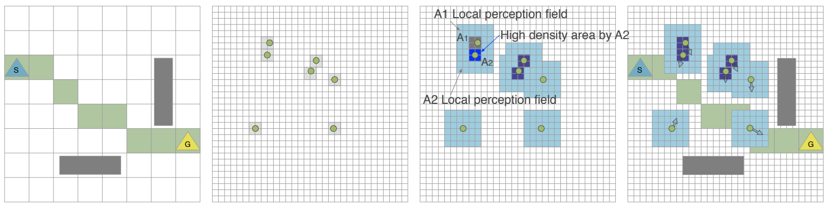
2 Related Work

A primary task in crowd simulation is to steer each agents towards its goal while avoiding collisions with other agents and obstacles. A popular global method to navigate agents to their goal positions is the A* algorithm, and several derivatives are proposed [19, 17]. The roadmap approach represents the connectivity of the free configuration space [16] that may be searched using different algorithms. In the various 'potential fields' approaches, a global field is used to represent the entire landscape with obstacles, and crowd elements move under the influence of these fields [1, 3]. Global path planning alone cannot model real crowds where each agent needs to react to dynamic factors for collision avoidance. Various techniques of the local planning for agent-based methods have been proposed, including rule based methods [14, 9], social force model [6, 11], and velocity models [2, 5], to just name a few. Generating sophisticated human reactive behaviors is of interest as well. Several studies mainly focused on psychological and emotional effects on individual behavior [13, 12, 20]. Some researches modeled the human cognition process in navigation [4, 8, 16].

Hughes proposed continuum dynamics based approach which handles global path finding and local collision avoidance as a single optimization problem [7]. Instead of computing each individual motion separately, crowds are represented as density fields, and individuals are guided toward goal by an evolving potential function on the density field. Treuille et al. brought the approach into a discretized particle representation, and provided compelling results for crowd phenomena [18]. Other works using continuum theory followed to generate an aggregate motion of crowds [10], and traffic simulation [15]. Compared to previous works on continuum based crowd models, our system supports individual behaviors of heterogeneous agents, and the performance is not bounded by the grid size of environment.

3 Virtual Environment Model

In our approach we divide the navigation into two major parts: global path planning and local steering. The global and local planning require a model of the environment that permits both planning modes to interact. We first introduce the model of virtual environment in this section.Our virtual environment is composed of a collection of 2D spaces, including a map field (§3.1), a world field (§3.2), and a set of local perception fields of agents (§3.3).



(a) (b) (c) (d)

Fig. 1: Environment model consists of (a) a map field, (b) a world field, and (c) local perception fields. (d) Combination of the three fields is shown.

3.1 Map field

A map field (*MF*) models terrain as a 2D grid, and static objects and topographic properties are encoded into cells of the grid. High level routing information is generated on the *MF* using a global path planning method and further details will be given in Section 4.1. Compared to world and local perception fields which will be discussed in following subsections, a relatively coarse grid is used because we only need overall connectivity information from point to point at this level of the map. Note that the simulation is not confined to use a 2D grid for the *MF*. Any form of maps can be used as long as we can perform a path planning algorithm on it and produce the geographic coordinates of paths and junctions. Therefore the simulation system can be augmented with other forms of maps, such as graph-based maps and roadmaps. Figure 1(a) illustrates a *MF* where static obstacles are represented as grey rectangles, and start and goal positions are marked with blue and yellow triangles, respectively. Planned paths that avoid the obstacles are drawn as a set of green squares.

3.2 World Field

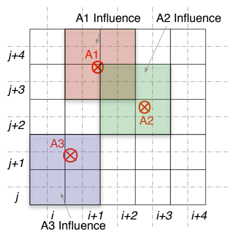


Fig. 2: Density values of cells with respect to location of agents

A world field (*WF*) is defined to handle agents and other dynamic objects. We represent a crowd in terms of density and flow velocity, as is standard for modeling pedestrian flow in continuum dynamics based crowd simulations. The *WF* is discretized as a regular 2D grid, and the continuous values of density and velocity are stored within the grid. An agent influences the density value ρ of the four closest cells to it. Figure 2 illustrates how density values are accumulated for each cell. Three agents A_1, A_2, and A_3 each have a 4-cell zone of influence (e.g. A_1 influencing cells $((i+1, j+3), (i+2, j+3), (i+1, j+4), (i+2, j+4))$). The ρ quantity accumulated in a cell owing to an agent is determined by the distance of the center of that cell from the location of the agent. The distance effect of agent A is modeled using a 2D Gaussian function. The density influence of A at (x_a, y_a) on a cell centered at (x, y) is:

$$\rho_a(x, y) = Ke^{-(\frac{(x-x_a)^2}{2\sigma_x^2} + \frac{(y-y_a)^2}{2\sigma_y^2})}$$

where coefficient K is the amplitude, and σ_x, σ_y are the x and y spreads of the blob. K, σ_x and σ_y can be controlled depending on a cell size, to generate an appropriate influence of an agent to the cell. The total density in cell (x, y) is then $\rho(x, y) = \Sigma_i \rho_i(x, y)$, where ρ_i denotes the density value of ith nearby agent. The velocity value $\bar{v}(x, y)$ is the sum of each agent's velocity scaled by its density and normalized by $\rho(x, y)$:

$$\bar{v}(x, y) = \frac{\Sigma_i \rho_i(x, y) v_i}{\rho(x, y)}.$$

Since the world field contains the information of dynamic factors, it is updated at each frame through the simulation. To keep track of each agent's position in reasonably fine detail, a relatively high resolution grid is used. Figure 1(a) and (b) show that both *MF* and *WF* covers the entire environment, however they are at different resolutions. In our experiment, we set a cell size of map field to be eight times that of the *WF*. The unit size of static obstacles and characters are considered to determine the cell size of map and world fields, respectively.

3.3 Local Perception Field

Local perception field (*LPF*) contains the measurement of cost for the locomotion action of an agent in a 2D grid form. We model the individual differences across agents by maintaining a *LPF* for each agent, and by allowing different agents to evaluate the cost by different criteria. In the real world, people interact with the surrounding environment but not with the whole world. Similar to this, an *LPF* only conveys the information of the immediate surrounding environment of each agent. Thus the perceivable range of an agent determines the size of the field. As an agent changes its position, the *LPF* moves along with it, keeping the center of the grid at the location of the agent, to reflect the immediate environment information.

The cost is quantified as a measure of discomfort arising from the density of the area and time-space factors. To measure discomfort σ_i for an agent A_i because of the proximity of other agents, all agents A_j within A_i's perception range are advanced by their velocity and transferred into density values as if the agents were there, over pre-defined amount of time steps τ. At each frame, density value ρ_j is accumulated into the four closest cells of A_j's location in A_i's local perception field: $\sigma_i(x,y) = \Sigma_{t=k}^{k+\tau} \Sigma_j \rho_j(x,y)$.

An example of arising discomfort for A_1 due to A_2's being within A_1's perception range at time step k is represented by dark blue cells in Figure 1(c). Similarly, discomfort aroused at A_1's location in A_2's local perception field is represented with grey cells. Time-space factors for the cost consists of path length, estimated travel time, and existence of static obstacles. Path length, travel time, and discomfort felt are linearly combined and weights for the terms are varied to reflect different individual principles in cost measurement. Finally, if a certain area is occupied by a static obstacle, its cost is set to infinity since agent cannot step into the area. The cost value $C_{x \to p}$ of a cell at p measured by an agent at location x is calculated as following:

$$C = \min(\alpha \int_p 1 ds + \beta \int_p 1 dt + \gamma \int_p \sigma dt,\ \delta), \tag{1}$$

where α, β and γ are weights for path length, travel time, and discomfort felt, respectively. δ is set to infinity for unavailable cells due to the static obstacles, otherwise set to 0. Figure 1 (d) shows a stack of three fields. As they move around the environment, agents collect information on the path, density, and velocity according to their current positions, and use these to decide final movement.

4 Crowd Navigation

Our hybrid approach of crowd navigation is decomposed into two levels. The first deals with the global path planning towards the goal, and the second addresses steering and local collision avoidance. We first describe them separately in §4.1 and §4.2, and show how these are integrated in a single framework in §4.3.

4.1 Global Path Planning

We perform global path planning using the A* algorithm in the *MF* to provide an agent with the preferred moving direction. Note that any global planners can replace the use of the A* algorithm as long as the geographic coordinates of paths and junctions are provided. For static features such as obstacles that do not change over time, the path planning is performed only once per agent at the beginning of the simulation. If crowd density is taken into consideration (as will be discussed later) and agent density is represented in the higher-resolution *WF* we need a way to compute the dynamic crowd density information in the *MF* for planning. We do this by aggregating the agents in the *WF* area covered by a *MF* cell. This aggregation includes both a count of the agents and an average of their directional vectors. In this case the A* algorithm will be invoked again as density conditions affect the planned path.

4.2 Local Planning and Steering

We now proceed to describe how the *LPF* is used for local steering to avoid collision with other agents. By definition, pedestrians prefer terrain with low cost while moving towards a destination. Therefore, agents will move in the direction perpendicular to the estimated cost, as there will be no advantage of moving along a line of constant cost. A potential function ϕ, which is a function to determine the optimal path, is found based on the cost C in *LPF*: $C = \|\nabla\phi(x)\|$. The potential field ϕ is assigned with the value of 0 inside a goal, and the other grid cell values are approximated by solving a finite difference approximation to the above equation outwards from the goal position. Note that the goal position inside the local perception field, called *local goal*, may not correspond to a final destination for an agent since the local perception field covers only a small portion of the environment. The way of identifying local goal will be described in the next subsection. Assuming a local goal is located, the final velocity v of an agent is decided by Equation 2, where $n_\theta = [cos(\theta), sin(\theta)]$ is the unit vector moving in direction θ, and rn_θ denotes an offset distance from the agent's location. The offset is given to have agents consider the velocity of the area in which they are moving into, but not of where they are currently at.

$$v = -\bar{v}(x + rn_\theta) \cdot n_\theta \cdot \frac{\nabla\phi(x)}{\nabla\|\phi(x)\|}. \tag{2}$$

The work of [18] is significantly bounded by the size and resolution of an environment grid, because the computation load grows proportionally to $O(N \log N)$ where N is the number of grid points. A common strategy for computation reduction is to employ a coarse grid to achieve real-time performance. The drawback of this approach is that with the concomitant increase in cell-size, multiple agents can reside in the same cell. This necessitates the use of pairwise comparison methods to check for collisions between agents, and the computation becomes a potential bottleneck in the simulation. An advantage of our approach is that it only solves the Eikonal equation on an *LPF* of small fixed size grid, the performance is not subject to the resolution and size of the environment. Since this *LPF* is represented at a high resolution grid, only one agent can occupy a cell and minimum distance enforcement becomes unnecessary.

4.3 Integration of Global and Local Planning

The integration of the global path planning and the local planning proceeds as follows. First, we perform the global path planning for each agent at the beginning of the simulation. As a result of this step, each agent is given the list of geometric coordinates of paths from its initial position to the final goal location. Second, each agent identifies a *local goal* within the range of its *LPF* with the given path information. Agent A_1 looks for the closest cell coordinates of the paths from its current position, which we call a *waypoint*. In Figure 3 the current waypoint for agent A_1 in the *MF* is shown as a green square. In the higher resolution *LPF* the local goal is set to the closest point to the center to the *MF* waypoint. Third, given the local goal for each agent, the continuum equation is solved to obtain the potential field for each *LPF* grid, and the final velocity for the agent motion is determined.

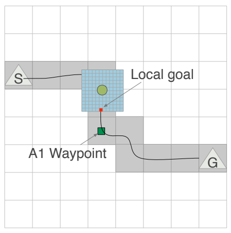


Fig. 3: Local goal selection on the paths.

Individualized behavior Step two of our above model is where behavioral variability is encoded with respect to individual agents. We provide three classes of different behaviors. The first class concerns an agent's preference on the path selection. Specifically, some agents may have greater aversion to congestion while the other may plan for the shortest path at the expense of other valuations for traversability, such as tolerance for going through a congested area. The second class takes into consideration the different levels of knowledge about the environment possessed by the agents. Some agents may know the entire map very well while the other may not. The third class of behaviors relates to group membership. Membership in specific groups may induce agents to behave differently than they would when isolated. For example, if one of their companion left behind, the rest tend to slow down or even stop to allow that agent to catch up. Individuals who do not belong to a group will travel only by their own decision. Details on the modeling of the three classes of behavior follows.

Path preference If dynamic factors like crowd density influence path choice of an agent, the global path has to be updated as needed. In such cases, if a waypoint is determined to be in congested area, the path finding algorithm described in Section 4.1 is recomputed using the agent's current position as the starting point. To determine the density at a particular waypoint at the *MF* scale, the system aggregates the corresponding *WF* occupancy information in the vicinity of the waypoint.

Knowledge level Agents in our system can have two levels of knowledge – full knowledge of a system-wide *MF*, and the naive case where the *MF* is ignored. The knowledgeable agent functions as described earlier with global *MF*-level path planning and *LPF*-level route planning. Naive agents that only have general goal positions have two choices for navigation: follow an authority figure which we call leader agent, or follow a crowd. Such agents use the *MF* to locate a leader or group to follow. Knowledgeable agents register their locations in the shared *MF*, and naive agents can query the *MF* to locate a leader nearby in the *MF* to follow. For crowd-following, the system employs the agent-density information described in §4.1. If a density value of a certain cell is higher than some threshold and those agents who contributed to the high density have similar moving orientation, this indicates that there is a coherent crowd to follow. For realism the agents only query nearby areas for leader or a crowd to follow. After targeting a leader or a group, following behavior of a dependent agent is simulated by setting its waypoint at the leader location or a center of the group.

Cohesive group movement The group aggregation factor, $\epsilon \int_m^p 1ds$ is introduced to equation 1 for the cohesive group movement. A center of mass m for a group is calculated and serves as a reference point for the group members to check their level of isolation from the group. As the value of m increases (it is far from its group), the cost of the path increases.

$$C_a = \max(\alpha \int_p 1ds + \beta \int_p 1dt + \gamma \int_p \varrho dt + \epsilon \int_m^p 1ds,\ \delta), \tag{3}$$

and ϵ is a weight for the group aggregation factor.

5 Implementation and Results

In this section, we discuss our system implementation and evaluation. Our crowd simulation is tested on a desktop with an Intel i7 2.67 GHz CPU, 12GB system memory, and a NVIDIA GTX295 graphics card.

5.1 Simulation Evaluation

A set of scenarios are used to test the system. The scenarios range from basic cases, testing the ability of the simulator to handle frequently encountered cases including congestion and oncoming threats, to large scale environment cases evaluating overall performance of the system. The scenarios include:
Congestion avoidance - A small number of agents start their navigation towards goal locations. While turning at a corner they confront unexpected congestion, a large number of agents block their optimal path to the goals.
Evacuation - A medium number of agents evacuate from an area. Some of them know where the exit is, and the rest of them are uninformed of the environment.
Cohesive movement - Several groups which consist of a small number of agents travel around a field. As a member of a group, each agent coordinates with other agents in the same group and show an aggregated motion as they move together.
Crossing individuals - From 1,000 to 20,000 agents with random goals cross a large flat area. The grid size is varied for the simulated environment, ranging from $1,024 \times 1,024$ to a very high resolution, 4096×4096.
People naturally steer around each other for collision avoidance without artificial enforcement of a minimum distance between each other (see Section 4.2).

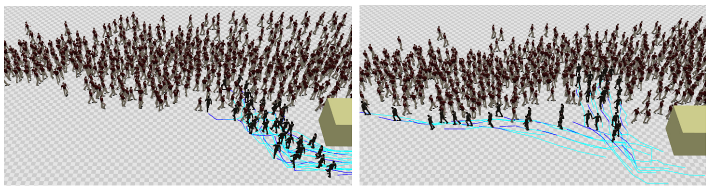

(a) (b)

Fig. 4: (a) All agents select the shortest path. (b) Each agent moves by its own preference on the paths.

Figure 4 shows our results for the *Congestion Avoidance* scenario. In Figure 4(a), all agents were set to prefer the shortest path by setting α to a high value in Equation 1. As can be seen, they surge into the same area without consideration to congestion. In Figure 4(b), in addition to shortest path preference, some agents were set to avoid congestion. As a result, some agents walk through the crowd while others circumnavigate the crowd as still moving toward their goal. Figure 5 (a) and 5(b) show the *Cohesive Movement* example. In Figure 5(a), eight groups of four-members each (color-coded for identification) move in clear clusters. The group cohesion behavior is excised, and no discernible grouping can be seen in Figure 5(b). Figure 5(c) is an example of our results running $10,000$ characters on a $2,048 \times 2,048$ grid. Smooth motion of the agents demonstrates that solving the Eikonal equation only on the immediate regions to the agent is sufficient to avoid collisions.

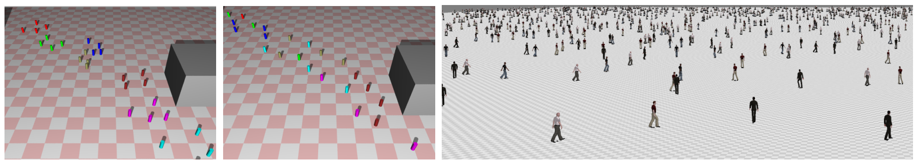


Fig. 5: (a) Agents show aggregate motion. (b) Agents move individually without concerning its membership. (c) A crowd of $10,000$ characters simulated on a $2,048 \times 2,048$ grid.

Table 1: The performance of our method on various configurations.

Number of characters		Grid Resolution		
		$1,024 \times 1,024$	$2,048 \times 2,048$	$4,096 \times 4,096$
$5,000$	Time/frame(ms)	20.77	28.87	47.82
	Fps	43	32	20
$10,000$	Time/frame(ms)	38.83	46.27	63.14
	Fps	23	20	14
$15,000$	Time/frame(ms)	57.3	65.68	86.87
	Fps	16	14	11

5.2 Parallel Implementation and Performance Results

We choose the NVIDIA's CUDA programming framework to be the target platform of our parallel implementation.

GPU-based Parallel Implementation Recall that our simulation has three levels of field, map, world and local perception field. Computation is decomposed following the fields organization. At the beginning of simulation, CPU-based computation is used to set up the initial positions and goal positions of the agents, and the topographical information of the environment. Global path planning for each agent is then performed on the CPU. All the information is passed to GPU, and a number of CUDA *threads* are created and they collaboratively compute the density and velocity values to fill the grid cells in *WF*. Once the global information is computed, the simulation is partitioned by agent such that one *block* of threads is assigned to build the *LPF* of each agent and to solve the related Eikonal equation.

Performance Results Our model runs at interactive rates up to $10,000$ characters on a $4,096 \times 4,096$ grid as shown in Table 1. Each character is assigned with a random initial and goal positions, and individual path preference with the *Crossing individuals* scenario in the experiment. The execution time per frame with various number of agents and grid cells is shown in Figure 6. The total execution time is linear both with the number of simulated characters and the size of

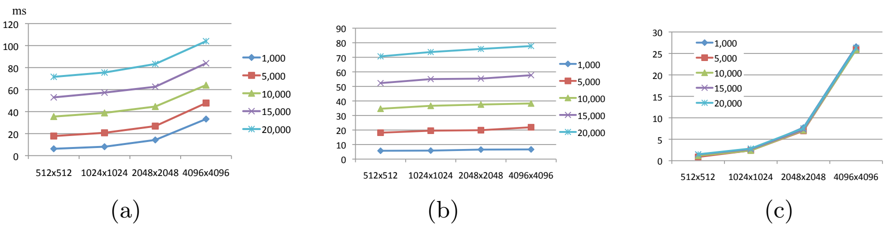


Fig. 6: (a) Simulation time per frame. (b) *LPF* time remains constant wrt. the size of grid. (c) *WF* time increases wrt. the size of grid.

environment grid (Figure 6(a)). Figure 6(b) shows *LPF* execution time remains approximately constant with the environment size. Since we solve the Eikonal equation on the fixed size of *LPF*, the execution is independent of the size of the environment. However, Figure 6(c) shows *WF* execution time to populate the density and velocity values for the environment increases exponentially to the size of environment grid, and becomes a potential bottleneck to the simulation.

6 Conclusion and Future Work

We present a hybrid crowd simulation system which leverages the benefits of both continuum-based and agent-based approaches. By modeling time-space information of surrounding environment of an agent as a local perception field, an agent's perception is linked to its evaluation of the cost of a particular terrain for traversal. The model captures the group dynamics in a large crowd as well as individual behaviors of each agents. From the performance results we show that our approach scales well on a large and high resolution simulated environment. Another possible advantage of our approach is that the simulation could be linked to any variety of options for the map field construction so that the continuum based simulation can be augmented with non-grid based paths.

We will continue our work in the following directions. First, we want to build a new global path planning components which can be implemented on GPUs. The current A* algorithm is designed for sequential execution. It might be the performance bottleneck of our system if many agents choose to dynamically update their global optimal path. Second, dynamic scoping of visual perception field could be applied. For the large relatively open environments, the agents would see further and this could have an effect on their behaviors. By using the dynamic size of local perception field, the load balancing among thread blocks will be an important issue on the parallel implementation as well. Third, we are currently working to simulate real world crowd scenarios, such as emergency evacuation. We plan to model a large region of a city and generate realistic agent behaviors for the simulation.

References

1. Arkin, R.C.: Integrating behavioral, perceptual, and world knowledge in reactive navigation. Robot. Auton. Syst. 6, 105–122 (June 1990)
2. van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: Robotics and Automation, 2008. ICRA 2008. IEEE Int. Conf. on. pp. 1928 –1935 (May 2008)
3. Chenney, S.: Flow tiles. In: Proc. of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation. pp. 233–242. SCA '04, Switzerland (2004)
4. Funge, J., Tu, X., Terzopoulos, D.: Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In: Proc. of ACM SIGGRAPH. pp. 29–38 (1999)
5. Guy, S.J., Chhugani, J., Kim, C., Satish, N., Lin, M., Manocha, D., Dubey, P.: Clearpath: highly parallel collision avoidance for multi-agent simulation. In: Proc. of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. pp. 177–187. SCA '09, ACM, USA (2009)
6. Helbing, D., Molnar, P.: Social force model for pedestrian dynamics. PHYSICAL REVIEW E 51, 4282 (1995)
7. Hughes, R.L.: A continuum theory for the flow of pedestrians. Transportation Research Part B: Methodological 36(6), 507 – 535 (2002)
8. Kapadia, M., Singh, S., Hewlett, W., Faloutsos, P.: Egocentric affordance fields in pedestrian steering. In: Proc. of the 2009 symposium on Interactive 3D graphics and games. pp. 215–223. I3D '09, ACM, USA (2009)
9. Loscos, C., Marchal, D., Meyer, A.: Intuitive crowd behavior in dense urban environments using local laws. In: Theory and Practice of Computer Graphics, 2003. Proc. pp. 122–129 (2003)
10. Narain, R., Golas, A., Curtis, S., Lin, M.C.: Aggregate dynamics for dense crowd simulation. ACM Trans. Graph. 28, 122:1–122:8 (December 2009)
11. Pelechano, N., Allbeck, J.M., Badler, N.I.: Controlling individual agents in high-density crowd simulation. In: Proc. of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation. pp. 99–108. SCA '07, Switzerland (2007)
12. Pelechano, N., Badler, N.: Modeling crowd and trained leader behavior during building evacuation. Computer Graphics and Applications, IEEE 26(6), 80–86 (2006)
13. Pelechano, N.: Crowd simulation incorporating agent psychological models, roles and communication. In: First Int. Wksp. on Crowd Simulation. pp. 21–30 (2005)
14. Reynolds, C.: Steering Behaviors for Autonomous Characters. In: Game Developers Conf. 1999 (1999)
15. Sewall, J., Wilkie, D., Merrell, P., Lin, M.C.: Continum traffic simulation. Computer Graphics Forum 29(2), 439–448 (2010)
16. Shao, W., Terzopoulos, D.: Autonomous pedestrians. Graphical Models 69(5-6), 246 – 274 (2007)
17. Sun, X., Koenig, S., Yeoh, W.: Generalized adaptive a*. In: Proc. of the 7th int. joint conf. on Autonomous agents and multiagent systems. vol. 1, pp. 469–476 (2008)
18. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. ACM Trans. Graph. 25(3), 1160–1168 (2006)
19. Trovato, K.I., Dorst, L.: Differential a*. IEEE Trans. on Knowl. and Data Eng. 14, 1218–1229 (November 2002)
20. Yeh, H., Curtis, S., Patil, S., van den Berg, J., Manocha, D., Lin, M.: Composite agents. In: Proc. of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. pp. 39–47. SCA '08, Switzerland (2008)