Crowd Self-Organization, Streaming and Short Path Smoothing

Stylianou Soteris
University of Cyprus
75 Kallipoleos Str.

Cyprus (CY-1678), Nicosia,

ssotos@ucy.ac.cy

Chrysanthou Yiorgos
University of Cyprus
75 Kallipoleos Str.

Cyprus (CY-1678), Nicosia,

yiorgos@ucy.ac.cy

ABSTRACT

Pedestrians implicitly cooperate by forming lanes inside dense crowd in order to facilitate flow and prevent complete passage blocking. Our aim is to re-enforce this self-organization phenomenon in dense crowd for the purpose of virtual crowd animation and navigation simplification. The mechanism of a *flow grid* is introduced to measure flow over an area. The flow grid is a perception mechanism of the surrounding area and favors dynamic lane formation (streams). It provides feedback to the navigation algorithm of the avatars, to enable them to choose a route that both meets their goal (wanted direction) and a trajectory that assists in self-organization of the crowd.

A very simplified yet fairly effective navigation method suitable for dense crowds is also presented. It demonstrates that self-organization of the avatars can help in simplifying local navigation. The method produces short distance, intermediate positions ahead in time and, as a post-processing step, smoothes them out before the avatar needs to use them.

Keywords
Crowd Navigation, Behavioral Navigation, Pedestrian Simulation.

1. INTRODUCTION

Reproducing believable crowds in virtual environments has always been a challenge. There are many research topics related to crowds. Such topics include character rendering, character animation, navigation, crowd simulation, collision avoidance, social behaviors and many more.

The focus of this paper is on the collective behavior of pedestrians, otherwise known as self-organization. A dense crowd of pedestrians, all trying to travel to unique and independent directions, creates a highly dynamic, changing environment. The pedestrians have to continuously confront many surrounding avatars whose navigation constantly changes. It is not possible to try to predict each opposing pedestrian's path, as that will probably change soon. Long distance path planning inside a dense crowd is futile. For this reason, the flow grid mechanism is introduced to simplify navigation and enable crowd self organization.

In the next section a summary of related work on navigation is presented. Section 3 presents the work on self-organizing crowd, followed by a description of the mechanisms needed to make this method feasible in Section 4. In Section 5 the results of the method are presented and finally in Section 6 a summary is presented and future work is discussed.

2. RELATED WORK

Crowd navigation deals with the problem of steering an avatar inside a large amount of static and moving obstacles (other avatars). The complexity of finding a suitable path increases as the density of moving avatars increases. There are three different approaches to solving the navigation problem. These are path planning, reactive navigation and behavioral navigation. Also of great interest to navigation is the computationally expensive collision detection and avoidance stage.

Most papers treat avatar navigation as a one person steering and path finding problem. The motion of an avatar from a starting point to a destination is treated as a global navigation problem. Various search algorithms try to find a complete path from one point to another. The most popular path planning algorithms are the Randomized Path Planner [Tsa03a], Probabilistic Road Maps as in [Kam04],[Sun05],[Sun04] and A* based Algorithms and variations such as [Cla87a],[Har68a]. In general path-planning algorithms are not a good option for dense crowd navigation because continuous re-planning would be needed due to the high number of dynamically moving objects. This of course would be inefficient.

Significant work has been carried in reactive navigation. Reactive navigation includes force field methods, rule based methods and XZT space methods. Force fields have been used by [Kam04], [Lam04], [Met03] for collision avoidance and smooth steering, but force fields are expensive to update. Rule based systems such as [Los03], [Tcc02],[Tcc01],[Nie03] have proven to be extremely fast as they can deal with thousands of avatars. The behaviors produced are limited only, by the number of rules applied. The Thalmann's also provide a good overview of previous work for groups in [Mus04]. Their work on emergent crowds is described. They model complex behaviors through a combination of attributes and rules, and through finite state machines. Their methods lack a global vision mechanism, thus forcing them to consider each neighboring avatar individually when making a decision. Our method will expand the rule based approach and add collective behavior by creating the mechanisms to read the collective behavior of other avatars in an area. XZT space methods exploit a space-time representation to better represent the movements of dynamic objects in space and time. The added T dimension allows for more accurate planning since it provides knowledge of the positions of all avatars ahead of time. Only a few experiments have been conducted with this method in [Feu00]. The same XZT space approach is also used by [Tsa03] in path planning for groups.

In behavioral navigation researchers try to simulate aspects of pedestrian behavior, mainly grouping, staying together, path following, yielding etc. Flocking [Rey87] is one such example. More recent work on behavioral navigation, include behavior maps [Los03], [Tcc01a], which cause pedestrians to adopt specific behavior for designated areas, according to the behavior map. Complete Sociological models have been proposed by [Sul02] and [Vil03]. Work on groups has also been carried out by [Kam04] and [Tsa03] , but it focuses on path finding and computation optimization for groups rather than behavioral aspects of group.

In crowd simulation and safety, [Sti00] presents extensive research with his thesis. Different behaviors of pedestrians in crowded areas are studied in detail. [Daa03] also reports on similar self-organization and other phenomena.

Collision detection/avoidance (CD) is one of the most expensive operations during navigation. Collision detection methods include bounding volume hierarchies, force fields, simple geometric tests, triangulation methods, proximity queries, occupancy maps, image-space methods and stochastic collision detection. A comprehensive review of CD methods can be found in the tutorial [Tes05].

3. SELF-ORGANIZING PEDESTRIANS

General Concept

Pedestrians follow other people when they travel in the same direction and need effort to overtake them. It is much easier to slow down and follow rather than walk into the opposing pedestrian stream when densely populated. Pedestrians need to be aware of any lane formation phenomena around them if they are to use the lanes to their benefit. Large coherent groups of people are very rare. Pedestrians form informal groups dynamically as they walk. They follow a bunch of people going in the same direction as them to avoid opposing streams [Mus04], [Daa03].

Overall Description of methodology

Initially a *flow grid* is constructed over the walk area. The flow grid measures the densities of pedestrians and their velocities at various directions. Thus the flow grid enables us to detect pedestrian density concentration and dominating direction of flow at any point in the walk area. The avatars use the flow grid information during navigation to select nearby areas that lie towards their final destination and have smaller opposing flow. In essence they avoid areas of high density or high opposing flow, thus lane formation takes place. The avatars continue to select nearby areas until they reach their final destination.

A high level path planning and areas/portals system is assumed on top of our system for navigating through a city. It is also assume that the avatars move on the XZ plane.

Measuring the Flows

Each avatar registers his position and velocity on the flow grid as soon as he moves to a new position. The velocities are separated into X and Z axis components. Positive and negative axis velocities are stored separately, Thus four velocity values are stored at each point, (+vx, -vx, +vz, -vz). Velocities are stored this way so that opposing velocities will not be canceled out.

The avatar's velocity and presence are distributed to the four neighboring grid points as shown in figure 1. The amount distributed to each corner depends on the avatars distance from that corner. The entire avatar density (1.0) is allocated to a corner when the avatar is exactly at that corner and is smoothly interpolated to zero as he moves away to the next corner.

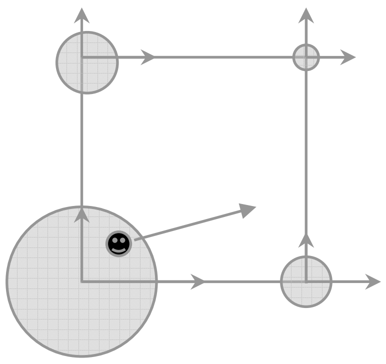

Figure 1. Each avatar is registered on the grid by distributing his density and velocity to the 4 neighboring points

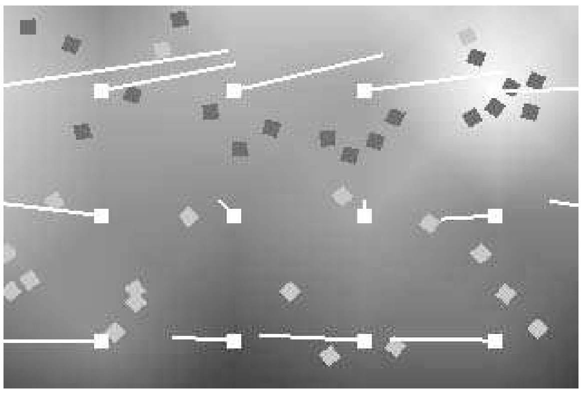

Figure 2. Light intensity shows the density distribution. White lines show the dominating direction of flow at each point. Dark grey avatars heading for the right border and light grey avatars for the left border.

Once all the avatars have registered on the grid, a complete picture of the densities and flows over the entire area is obtained, as shown in Figure 2. Higher intensity grey areas indicate more people near that grid location.

Later on, the flow grid is used to extract information for navigation purposes. Densities and velocities are interpolated between grid positions when information is needed at any in-between point.

Using the Flow Grid to Navigate

A higher layer that assigns long distance destinations to each avatar has been assumed, e.g. [Sty04]. Then the steering towards that destination must be performed. For example an avatar is told to steer from one end of a road or square to another end by the higher layer. The avatar has to navigate to that destination. He does so by selecting intermediate local targets just a few meters ahead of him. By consulting the flow grid at regular intervals the avatar chooses to head for the area with smaller density and smaller opposing flow. To help the avatar decide which area is best, a special weight formula has been constructed. This formula is explained in the box below:

Weight = (1 + D) * (1 + AngleDiff (T, F))

D = density at spot

T = vector showing direction towards target pos

F = vector showing direction of flow at spot

AngleDiff = Angle difference between 2 vectors in radians

The weight is a product of the density at that spot and the angle difference between the direction towards the avatar's target and the dominating direction of flow on the flow grid.

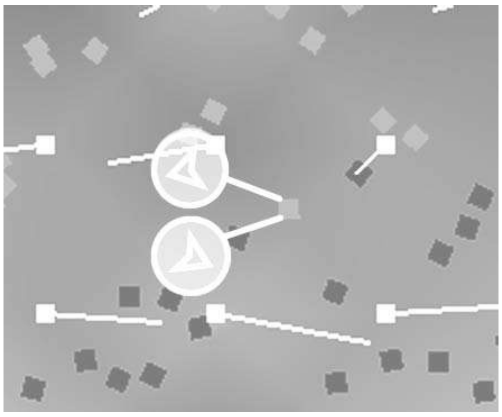

Figure 3. The arrows in circle demonstrate the dominating flow on the *flow grid* at that position.

Two weights are calculated each time the avatar needs to find a new intermediate destination. Each weight lies approximately 2 meters ahead of each pedestrian and to his left and right. The spot with the lowest weight is chosen as a temporary local target as shown in Figure 3.

The weight has been formulated empirically and it gives a nice measurement of the density and opposing flow at any spot.

The avatar continues to steer to the temporary target until a new temporary target is calculated in a short time interval. The grid takes care of the lane formation behavior of the avatars. The steering and collision avoidance method is presented next.

4. LOCAL STEERING AND SMOOTHING

About steering in densely populated areas

Pedestrians walking in densely populated areas change direction to avoid others all the time. Checking for long free paths is not very useful. Even if there is a free path along a direction, it can very easily be claimed by other pedestrians and soon become unavailable. A simple, but slightly deferent than usual, approach has been used by our system.

Description of steering algorithm

A discrete occupancy map is used for collision avoidance. The avatar maintains a list of six positions at any time. Each position is no more than one occupancy cell away from the previous position. At all times the actual avatar position is interpolated between positions 2 and 3. When the avatar reaches position 3, the first position in the list is discarded and a new position is added to the end of the list. The avatar is looking only one occupancy cell around him for a new position. For this reason, a very sharp turn may be needed. Smoothing is used to minimize sharp turns, as a post processing step. Essentially, the first 4 positions are used for curve interpolation and the last 2 are used for path smoothing

Position interpolation

Interpolation allows us to perform path planning only when needed. Practically path planning occurs approximately 3 times per second. The occupancy map cell size is 33cm and the average avatar speed is 100cm per second, so if avatar positions in the list are to be adjacent, position searching will occur 3 times per second. All intermediate positions are interpolated using a Catmul-Rom curve. This is an excellent optimization since simple interpolation is the only operation performed for most frames and path planning is postponed until it needs to be performed.

Next position search

Only the cells around the avatar are checked. Long free paths are not checked, instead, free cells in front of the avatar are checked. Figure 4 shows the search area when looking for a new position. When the avatar runs into dense crowd, he shouldn't turn back where he came from, as it would look unnatural. For this reason the search distance is being reduced when the rear of the avatar is being searched.

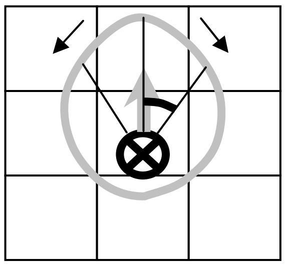

Figure 4. The search area for possible positions is shown here. The search starts from the top and checks left and right at an increasing angle and a reducing distance until an empty cell is found.

Path post-smoothing

Due to the limited search space ahead of the avatar and the densely populated environment, sharp turns can occur quite frequently. For this reason a path smoothing step is performed as post-processing. The curve is smoothed out by moving the in-between position (5th step) to a better location if that is still empty in the occupancy map. Figure 5 demonstrates a smoothing step. Position 5 is moved towards the line formed by positions 4 and 6 in order to minimize the turn angle.

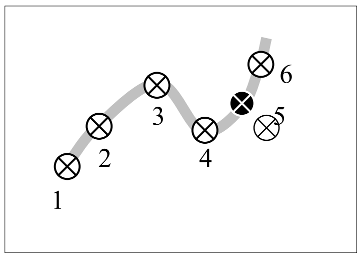

Figure 5. The 6 positions maintained by the avatar. The first 4 are used for curve interpolation. Position 5 is canceled and moved to a better position (black spot) during smoothing.

Each position relates to the occupancy map at the time the position was taken. Because of smoothing, positions taken at past time need to be changed. Occupancy maps for the previous time-steps need to be maintained as well to validate whether such a change is possible. Therefore 2 occupancy maps are maintained, one for the step at position 5 and one for the new position 6. If the new desired position at step 5 is not available on the occupancy map then smoothing does not occur. It would be possible to keep more positions for smoothing, which in turn would require more occupancy maps.

5. RESULTS

Various configurations of crossing and parallel crowd streams have been tested. The results of those tests as well as the local steering results are presented in this section.

Two Parallel But Opposing Streams

The first test was composed of two opposing crowd streams. The avatars are released from random points on one end of the walk area and navigate to another random point on the opposing side of the area. As the two opposing streams meet they start to formulate streams and lane formation becomes more stable after a while (Figure 6). Avatars may need to cross opposing streams if they need to head for a different destination point.

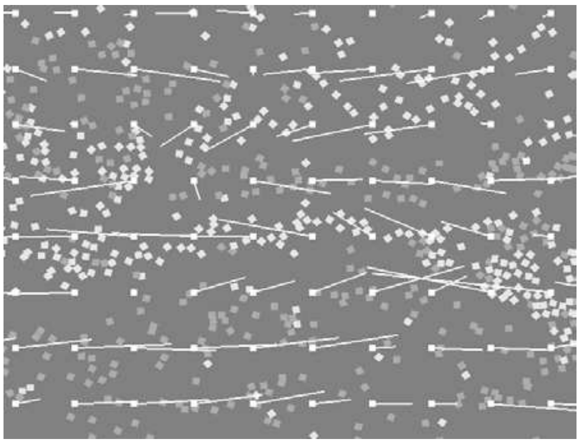

Figure 6. Lane formation between two opposing crowd streams. The dominating direction at each point on the grid superimposed on top.

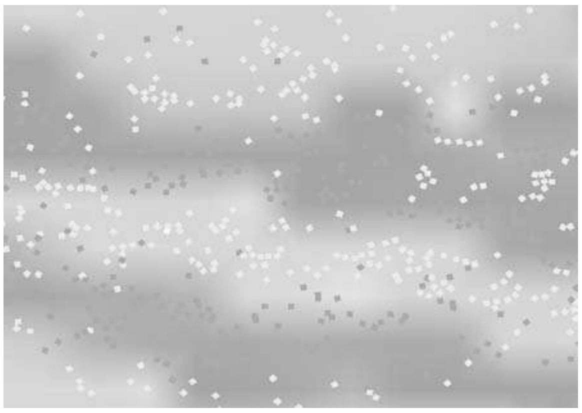

Figure 7. Lane formation between two opposing crowd streams. The flow direction graph is under-imposed.

To better visualize the lanes formulated, a flow direction graph is rendered (Figure 7). The direction of flow at each point on the grid is mapped to a color using a grey color scheme.

Two Crossing Crowd Streams

The results are different for crossing streams. Constant lane formation is not possible, as the two streams need to cross each other. What happens is vertical/horizontal alternation of streams at an area. The momentary lane formation for one direction is followed by alternation once the avatars from the opposing direction start to gather up. The dominating stream blocks the avatars traveling in a vertical direction to the stream, forcing them to slow down. Once the waiting avatars start to gather up, the stream changes direction. Figure 8 shows two crossing streams of avatars.

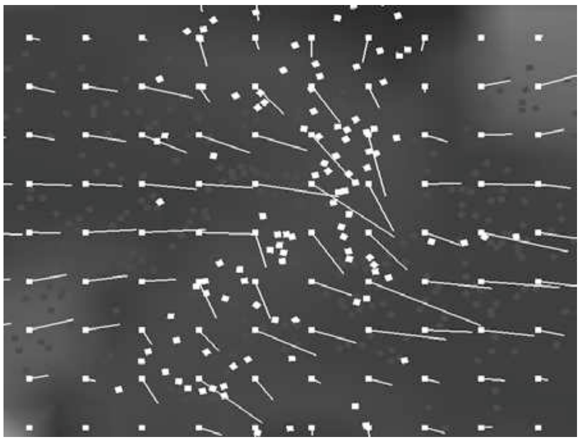

Figure 8. Two crossing crowd streams. Lanes are formed only momentarily, temporarily slowing down avatars that try to cross it.

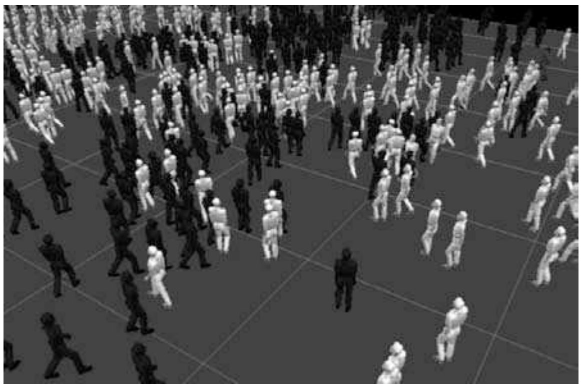

Figure 9. Two crossing pedestrian streams. The avatars are rendered as impostors. Different colors are used to distinguish avatars belonging to different streams.

It can be seen in Figure 9 that the two crossing streams of avatars mix much more than the previous test case. The avatars in white color need to travel in a perpendicular direction to avatars in grey color. Crossing streams are much harder to distinguish.

Four Crossing Crowd Streams

As Daamen and Hoogendoorn [Daa03] state the resulting self-organization at crossings is just chaos. It has been observed that here too, lane formation is only momentary. It only happens for very short moments, as there are four crowd streams now competing for the same area. The flow measurements on the grid do favor only one direction, the dominating direction at each area. Thus some avatars choose and some avoid an area, which causes the momentary lane formation. Figure 10 demonstrates the resulting flows at a crossing.

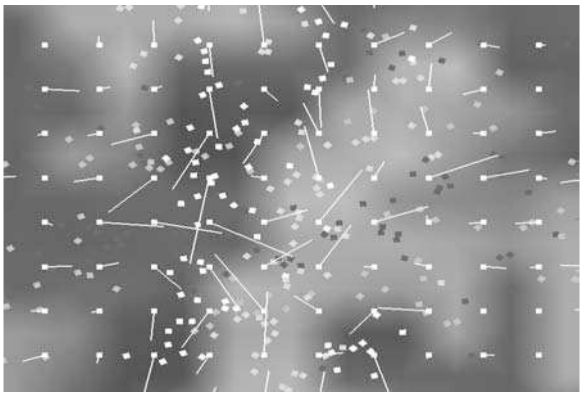

Figure 10. Four crossing crowd streams competing for the same areas. Lanes are formed only momentarily, mostly its only chaos.

Resulting Paths

Some resulting paths are shown in Figures 11 and 12. Figure 11 shows examples of horizontal paths inside two horizontally opposing streams. Figure 12 shows example paths of perpendicularly crossing streams.

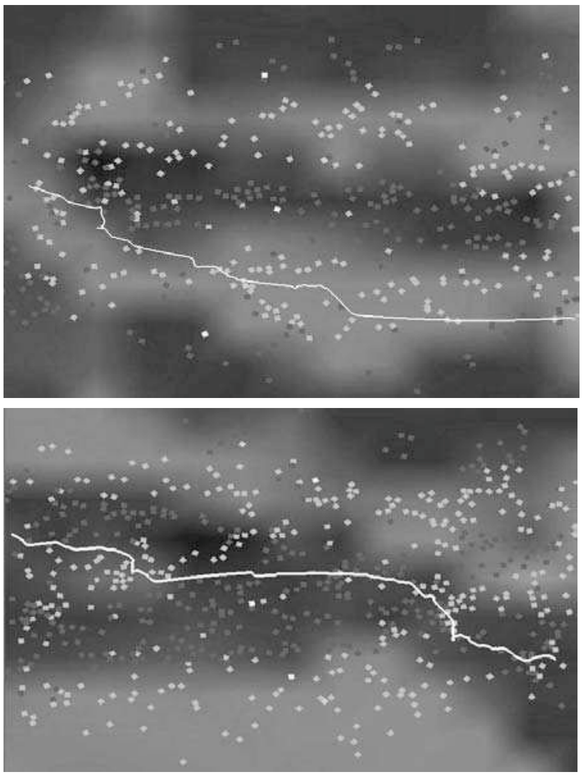

Figure 11. Resulting paths of parallel but opposing streams

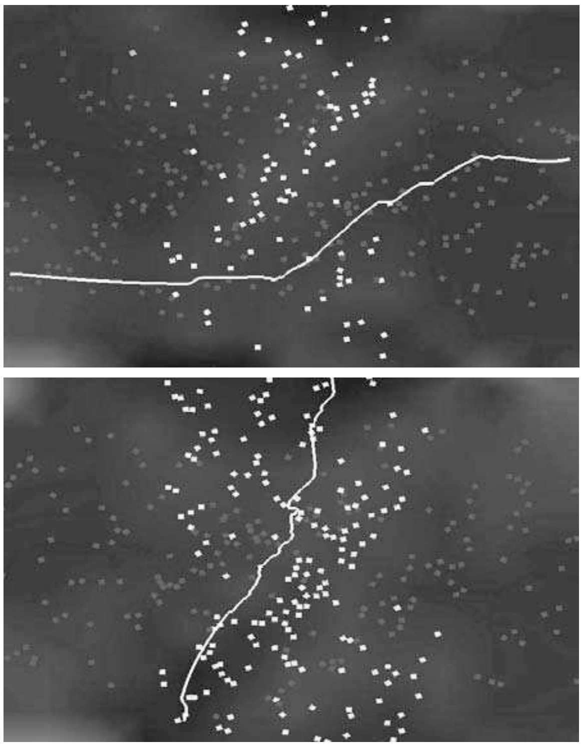

Figure 12. Resulting paths of perpendicularly crossing streams

The granularity of the flow grid affects the lane formation. The lanes become thicker when the grid is coarse, and thinner when the grid is finer. A granularity of 5m to 10m between flow grid points has been found to produce pleasing results.

Performance

The tests have been performed on a Pentium 4, 3Ghz machine with 512Mb RAM and Geforce 5600 with un-optimized code and are running at interactive frame rates for up to 2000 avatars. The simulation area is an open space of approximately 3000 square meters.

The performance cost for the flow grid is minimal. It lies between 1-2% of the navigation performance cost. The relationship of the number of avatars and the flow grid cost is linear. The size of the area and the density of the grid do not affect the performance cost of the flow grid. It is important though to point out that the resulting crowd streams allow for higher densities of avatars to flow through an area, whereas without the flow grid they would jam. Table 1 shows that approximately 20% to 25% more avatars can flow through an area before the crowd jam density limit is reached.

Area in sq.meters	Max. Number of avatars without streaming (approx.)	Max. Number of avatars with streaming (approx.)
3000	~600	~800
5200	~850	~1200
20800	~2500	~3800

Table 1. Improved Density of avatars. Approximate crowd jam limits for different area sizes.

Number of avatars	Average Algorithm Frame Time (msec)	Number of avatars	Average Algorithm Frame Time (msec)
185	11.3	760	48.4
275	16.5	830	55.1
340	21.1	900	59.9
430	26.9	1020	65.4
570	35.2	1230	78.4
670	42.03	1320	86.5

Table 2. Local Navigation performance cost.

The local steering is the most costly part of the algorithm since it includes local collision avoidance. It has been observed that the critical factor in the performance of local steering is the density of the avatars. Table 2 shows the local collision avoidance cost. The performance of navigation is fairly linear (Figure 13) until near maximum capacity is reached. Once the density of avatars becomes too high the cost of collision avoidance rises exponentially due to the high density and continues collisions between avatars. The performance cost for collision avoidance is significantly lower than other methods because each avatar only searches for a step every few frames and only looks at the neighboring cells on the occupancy map around him.

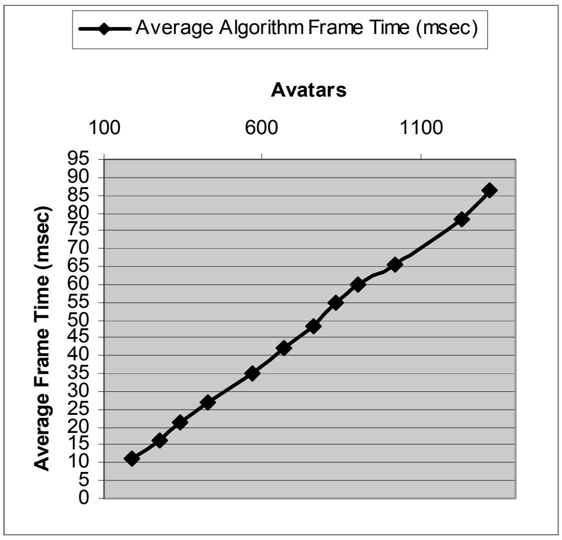

Figure 13. The graph for Table 2. Performance of local steering algorithm.

6. DISCUSION

This paper presented a method for lane formation behavior in crowds and a method for quick steering inside dense crowd. The method has been visually evaluated and it gives pleasing walking crowd results.

The flow grid gives an overall impression of what kind of pedestrian traffic exists in that area. It can easily be used with more complex path planning algorithms for a more detailed path planning approach. The simple path selection algorithm presented here only examines two areas ahead of the avatar, but it has been found to be adequate for dense crowd navigation. Congested areas can also be detected from the grid since they will have high density and low flow.

One important benefit of the steering algorithm is that there is no collision avoidance cost while the avatar positions are being interpolated from one pre-planned step to the next. The collision avoidance costs are thus significantly reduced. Collision avoidance is only performed each time a new step needs to be planned.

The steering method is suitable for a dense pedestrian environment. It cannot operate alone. It needs to be directed towards short distance intermediate waypoints. The intermediate waypoints can be extracted from the flow grid or a similar method. The avatar path list can be extended to include more points and smoothing can be applied to more points in the path list to produce smoother paths.

7. ACKNOWLEDGMENTS

Special thanks to Franco Tecchia for permitting use of his impostor rendering library.

8. REFERENCES

[Cla87] Clarcson, k. (1987). Approximation algorithms for shortest path motion planning. In Proceedings 19th Annu. ACM Symposium of Theory Computation.

[Daa03] Daamen W. and H. S.P. (2003). Experimental Research of Pedestrian Walking Behavior. Louisiana, Louisiana Transport Research Center.

[Feu00] Feurtey, F. (2000). Simulating The Collision Avoidance Behavior of Pedestrians. Dept. of Electronic Engineering. Tokyo, University of Tokyo. MSc: 53.

[Har68] P. E. Hart, N. J. Nilsson, et al. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." IEEE Transactions on Systems Science and Cybernetics(SSC4 (2)): 100-107.

[Kam04] A. Kamphuis and M. H. Overmars (2004). Finding Paths for Coherent Groups using Clearance. Eurographics/ACM SIGGRAPH Symposium on Computer Animation.

[Lam04] LAMARCHE F and D. S (2004). "Crowd of virtual humans: a new approach for real time navigation in complex and structured environments." Computer Graphics Forum 23(3): 509-518.

[Los03] Celine Loscos, David Marchal, et al. (2003). Intuitive Crowd Behaviour in Dense Urban Environments using Local Laws. Proceedings of the Theory and Practice of Computer Graphics 2003, IEEE Computer Society.

[Met03] Ronald, A. M. and K. H. Jessica (2003). Reactive Pedestrian Path Following from Examples. Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003), IEEE Computer Society.

[Mus04] Soraia Raupp Musse, Branislav Ulicny, et al. (2004). Chapter 14 summary of Handbook of Virtual Humans. Groups and Crowd Simulation. D. T. Nadia Magnenat-Thalmann.

[Nie03] C. Niederberger and M. Gross (2003). Hierarchical and Heterogenous Reactive Agents for Real-Time Applications. Eurographics 2003, Granada, Spain.

[Rey87] Reynolds, C. W. (1987). "Flocks, herds, and schools: A distributed behavioral model." In M. C. Stone, editor, Computer Graphics (SIGGRAPH '87 Proceedings), urban walking environments. Journal of Planning Literature: 25-34.

[Sti00] Still, K. (2000). Crowd Dynamics. Department of Mathematics. Warwick, Warwick. PhD.

[Sty04] Stylianou, S, Fyrillas, M and Chrysanthou, Y (2004). "Scalable Pedestrian Simulation for Virtual Cities". ACM VRST 2004, Hong Kong, November 2004.

[Sul02] C. O'Sullivan, J. C., H. Vilhjálmsson, J. Dingliana, S. Dobbyn, B. McNamee, C. Peters, and T. Giang (2002). "Levels of Detail for Crowds and Groups." Computer Graphics Forum, Volume 21 Issue 4: 733.

[Sun04] Sung, M., M. Gleicher, et al. (2004). Scalable behaviors for crowd simulation. EUROGRAPHICS 2004.

[Sun05] Mankyu Sung, Lucas Kovar, et al. (2005). "Fast and accurate goal-directed motion synthesis for crowds". Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2005).

[Tcc01] Franco Tecchia, Celine Loscos, et al. (2001). "Agent Behaviour Simulator (ABS): A Platform for Urban Behaviour Development". ACM/EG Games Technology Conference.

[Tcc02] Franco Tecchia, Celine Loscos, et al. (2002). "Visualizing Crowds in Real-Time." Computer Graphics forum Volume 21(Number 4): pages 753-765.

[Tes05] Matthias Teschner, Bruno Heidelberger, et al. (2005). Tutorial: Collision Handling in Dynamic Simulation Environments. Dublin, Ireland, Eurographics 05.

[Tsa03] Tsai-Yen Li and H.-C. Chou (2003). Motion planning for a crowd of robots. International Conference on Robotics and Automation (ICRA). San Diego, CA, IEEE Press.

[Vil03] Marta Becker Villamil, Luiz Paolo Luna de Oliveira, et al. (2003). A Model for Grouping Virtual Individuals Based on Social Behaviors. Intelligent Virtual Agents 2003. Irsee, Germany