# Levels of Real-Time Crowd Navigation Behaviour in Urban Environments

M. Haciomeroglu[1], R. G. Laycock[1] and A. M. Day[1]

[1]School of Computing Sciences, University of East Anglia, UK

**Abstract**

*Continuously increasing interest in real-time crowd simulations motivates researchers from multiple disciplines to investigate more realistic and efficient simulations. Often real-time crowd simulation systems are developed to enhance a user's experience. Therefore, in these applications, most of the computer processing power assigned to the crowd simulation should be given to the agents that will clearly gain a user's attention. This paper investigates methods to improve the computational efficiency of the simulation while not effecting the user's experience. For this reason six dynamically sized areas are defined according to the camera's position and direction. Three types of motion algorithms with different complexities are defined to be used in the different simulation areas. The proposed technique is used to simulate 10000 pedestrians in a $4km^2$ urban environment and is shown to offer more than 10 times improvement in computational performance.*

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.8]: Applications—

## 1. Introduction

In real-time crowd simulations it is typically the case that the behaviour for all individuals in a crowd will be computed irrespective of the current point of interest. Of course this is not an optimal strategy nor is it especially desirable for real-time applications. A few attempts have been undertaken that involve the reduction of the computational cost for simulating those agents that are invisible or too far away from the camera. However, as discussed in Section 2 these techniques do not sufficiently take advantage of a user's inability to observe smooth locomotion, if the agents are too far from the camera. Ideally no computation should be conducted for agents outside of the visible area, but this will result with the agents that are outside of the visible area being completely stopped. This can lead to undesirable artifacts, since the user's camera movements are unpredictable, a user could potentially leave an area and when returning to that area notice an inconsistency with the agents' positions. To alleviate this problem a very basic intelligence could be performed for those agents close to the camera but currently outside of the view frustum. However, this also has disadvantages, since a reduced behaviour system may not be able to avoid inter-agent collisions and when the camera turns to view those areas currently out of view a user may see the agents immediately start to escape from those collisions. Therefore even though the agent is invisible a computationally efficient and sufficiently intelligent motion technique is needed to avoid collisions.

Even if the virtual agent is inside the view frustum it can be far away from the camera and consequently a user can not perceive collision artifacts. The proposed system in this paper is able to use that fact by introducing an unsophisticated steering technique which immitates the physically accurate steering technique when viewed from a distance. This is achieved by altering the agent's goal points by an amount that is changing dynamically depending on the density and direction of the pedestrians.

Specifically where to use these locomotion algorithms, which have different levels of quality and computational complexity, is another topic of this paper. In this paper six different behaviour areas are introduced. In each area a different combination of the locomotion techniques are used in order to improve the computational efficiency while not effecting the user experience. The different behaviour areas have variable size based upon the height of the camera. Indeed if the camera leaves eye level and starts flying, a user is able to see the virtual agents' locomotion more clearly, even if the camera and the virtual agent distances have increased. The presented approach also takes into account this fact in order to maintain the quality of the simulation.

This paper is organized as follows; in Section 2 the previous related work is discussed, in Section 3 two separate pedestrian spatial indexing techniques are compared, in Section 3.1 different behaviour areas are explained, in Section 3.2 three locomotion techniques are presented and in Section 4 this work is concluded and possible future work is discussed.

## 2. Previous Work

In 1997 Musse and Thalmann introduced a multi-resolution collision detection procedure [MT97] into their group based virtual crowd simulation. They used two types of collision detection and response algorithm, one is cheap to perform and the other is more computationally expensive. They choose between the two depending on the distance to the camera and concluded that multi-resolution collision detection algorithms can improve the performance while not introducing clearly observable collisions. Building upon this work Pettré et al. [PdHCM*06] introduced Levels of Simulation (LOS) which is a more sophisticated multi-resolution collision detection procedure. They divided the environment into three areas; visible, visible but far away and invisible. In the visible and nearby locations they used Reynold's steering behaviours [Rey99] and linear steering elsewhere. However, their LOS technique does not take into account the pedestrian densities and average flow directions when simulating agents outside of the view frustum. This leads to an unrealistic distribution of virtual agents when the pedestrians switch from being invisible to becoming visible.

Chenney et al. introduced proxy simulation [CAD01] a technique in which cars outside of the visible area are updated less frequently and the speed of the cars are set based on the pre-calculated full simulation. (In a full simulation every location is deemed visible and the highest level of behaviour simulation is performed for all agents). However, the proxy simulation does not solve the problem of levels of simulation inside the view frustum. Furthermore, in densely populated narrow areas, updating the pedestrians less frequently can lead to obstacle collision. Proxy simulation is ideal for long straight roads with low numbers of virtual entities (e.g. cars). Recently Marini et al. [MYMT07] proposed a simulation with two different steering mechanisms; one for close agents and the other for agents that are further away according to the camera position. This approach does not fully take advantage of the invisible agents since the locomotion techniques switches only based on the distance.

Several techniques exploiting the idea of levels of artificial intelligence have been discussed in the literature, [OCV*02]. However, these systems are designed for reducing the computational complexity of higher levels of behaviours, such as buying a ticket or waiting for a train etc. This paper investigates the locomotion behaviour levels according to camera orientation for a low level behaviour system. The main focus is to develop several locomotion techniques which have varying complexities and to use these techniques in appropriate places in order to improve the computational performance while not detracting from the user experience in a 3D virtual urban environment.

## 3. Pedestrian Indexing (Graph vs Grid)

In real-time crowd simulations, researchers have widely adopted the regular grid data structure to determine quickly a given agent's neighbouring agents to permit efficient collision avoidance strategies. In contrast, the system presented in this paper uses another technique called navigation graph based indexing (NGBI); the concept was previously outlined in [PGT08], however, a comparison between the two approaches was omitted. In this technique every agent has a path consisting of navigation graph nodes; the graph nodes represent physical edges for the agents to navigate. While agents traverse their paths node by node they update their current node with their existence.

For this simulation, the pedestrians are simulated in a $4km^2$ urban environment and traverse a navigation graph consisting of 9932 graph nodes, which represent the pavements. The neighbouring agents for a pedestrian are determined by checking the pedestrians on a precomputed list of neighbouring edges. The neighbouring edges are found via a depth first traversal of the edges, while their distance from the current node is less than the maximum visibility, $10m$, of a pedestrian. This technique was compared to a regular grid, with $10m^2$ grid cells, and a 8% improvement in computational performance was achieved with 10,000 pedestrians being simulated. Whilst this approach is more efficient it has been adopted for agent navigation since it is able to realistically distribute the pedestrians in visually less significant places, as discussed in Section 3.2.

### 3.1. Determining the behaviour level for each individual

In the proposed simulation there are 6 different behaviour areas defined: A, B1, B2, C1, C2 and C3. Each area has its own combination of pedestrian locomotion behaviours. Each pedestrian is assigned to one of these behaviour areas by using Table 1, where $P_i d$ denotes the distance from the camera to the $i^{th}$ pedestrian. All 6 behaviour areas and a sample camera direction are illustrated in Figure 1.

Euclidean distance thresholds of $th_1$ and $th_2$ are shown in Figure 1. These variables change dynamically depending on the camera height ($h$). When the camera starts flying the user is able to see more pedestrians and their trajectories, hence these distances should be extended based on the camera height. $th_1$ is calculated using Equation 1, where $h$ is the

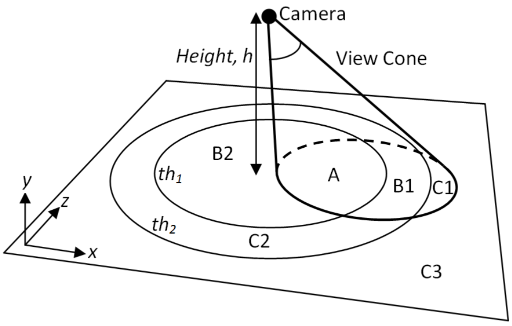


**Figure 1:** *3D view of the behaviour areas.*

| | Visible | $P_id < th_1$ | $P_id < th_2$ |
|---|---|---|---|
| A | true | true | true |
| B1 | true | false | true |
| B2 | false | true | true |
| C1 | true | false | false |
| C2 | false | false | true |
| C3 | false | false | false |

**Table 1:** *Rules indicating in which behaviour area an agent should reside*

height of the camera. In fact Equation 1 is a shifted version of the ellipse equation where $a$, equal to 125, represents the semimajor axis and $b$, equal to 100, represents the semiminor axis. $th_2$ is calculated by using Equation 1 with the values of $a$ equal to 250 and $b$ equal to 200.

$$th_{\{1,2\}} = b\sqrt{(1-\frac{(h-a)^2}{a^2})} \qquad (1)$$

### 3.2. Levels of navigation behaviour

In the proposed system there are three levels of locomotion behaviour: level 1, level 2 and level 3.

#### 3.2.1. Locomotion level 1

This level of locomotion is designed for efficiency and is used for pedestrians located in areas C1,C2 and C3 in Figure 1. In the current system C1 and C2 are separated only for a clearer definition, whereas C3's agents are updated less frequently. The significant requirement of locomotion level 1 concerns the distribution of the pedestrians realistically along the graph nodes. This is because pedestrians in areas C1, C2 and C3 are either outside the view frustum or are too far away from the camera to properly perceive collisions amongst them (as discussed in Section 3.1). Therefore, an efficient locomotion technique has to be designed for those areas, taking into consideration that when the user changes their focus to the places where previously level 1 locomotion was used by pedestrians, the user should not be aware that a computationally more efficient locomotion technique had been used. This means that no collisions should be perceived and the agents should travel distances at the same speed as if they had been simulated with complete steering and collision avoidance.

In locomotion level 1, only a goal seeking behaviour is simulated. However, if all the pedestrians on an edge were directed towards the end point then this would cause the pedestrians to move in a line. If the pedestrians became the focus of a user they may quickly separate to avoid collisions and a user will easily observe this artefact. Therefore, the system alters the pedestrian's goal points based on the density and average direction of pedestrians on each graph node. A sample navigation graph node (edge) $P_1 - P_2$ and its maximum clearance gates $A_1 - B_1$ and $A_2 - B_2$ are illustrated in Figure 2. The maximum clearance gates can be determined by using a maximum cylindrical area method [PLT05], however, in this simulation the free spaces may have more than one navigation graph node, (these represent the edge of the pavements in the urban environment). Therefore, all the graph nodes are sampled by casting perpendicular rays from positions along the associated edge and directed into both half spaces of the edge. The sampled rays terminate when they come into contact with a building and the rays with the shortest length from both sides of the edge are recorded to determine the size of the gates. These gates are used to alter a pedestrian's sub-goal point which was only permitted to be equal to $P_1$ or $P_2$ before. The line segments $A'_1 - B'_1$ and $A'_2 - B'_2$ in Figure 2 are determined by using both the density $\rho$, (which is equal to the number of agents on the edge, $N$, divided by the edge's length in metres), and the average direction of pedestrian movements along the edge, $mD$. While pedestrians navigate the graph nodes they will update the value of $mD$ for the graph nodes that they pass through. For instance, if they are travelling towards $P_1$ they will add 1 to $mD$, and if they are travelling towards $P_2$ they will subtract 1 from $mD$. The ratio $r$ of the line segments $A_1 - B_1$ and $A'_1 - B'_1$ is calculated by using Equation 2. This simulation uses graph based indexing and therefore, determining the average density of a graph node does not require any calculation. As the density increases and the average movement direction tends to zero, the longer the line segments $A'_1 - B'_1$ and $A'_2 - B'_2$ will become. If the current $mD$ is zero and there is a high density on a node this means that equal numbers of pedestrians are travelling in each direction and therefore in the interest of collision avoidance the pedestrians may disperse widely.

By using the value $r$ calculated in Equation 2 the pedestrians' speeds can be decreased up to half of their intended speed. This ensures that pedestrians on graph nodes that have a high $r$ value will slow down more in order to keep consistent to the full simulation. This is because without the decrease in speed, pedestrians will reach their destination di-

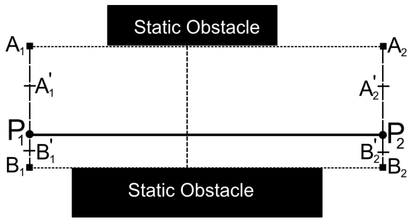


**Figure 2:** *The two gates of an edge associated with a graph node.*

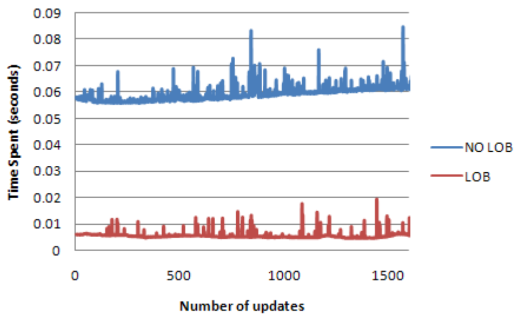


**Figure 3:** *A graph comparing the full simulation with the proposed levels of behaviour technique.*

rectly, instead of taking longer due to the most direct route being hindered by other pedestrians that should be avoided.

$$r = min\{(1 + \frac{(N - \|mD\|)}{N}) \times \frac{\rho}{2}, 1\} \qquad (2)$$

#### 3.2.2. Locomotion level 2

Locomotion level 2 is used in areas A, B1 and B2 illustrated in Figure 1. This locomotion level adapts the dynamic obstacle avoidance behaviour presented in [Rey99]. The extension is applied in area B1 where the maximum visible area of a pedestrian decreases from $10m$ to $2m$ while the distance to the camera increases.

#### 3.2.3. Locomotion level 3

This level of locomotion is used in area A. By only using locomotion level 2 it is not guaranteed that pedestrians will obtain collision free movements in densely populated places. Every pedestrian in area A will also check the other agents that are in close proximity and in front of them and from this information generate a push vector. The push vector is summed with the direction vector, calculated using locomotion level 2, to form the final movement vector. The push behaviour introduces a decrease in pedestrian speeds in dense areas, since the push vectors may decrease the scale of the direction vector. Furthermore the push vector allows the pedestrians to wait for a short period of time before moving freely.

## 4. Results, Conclusion and Future Work

To evaluate the performance gain afforded by the levels of behaviour techniques described in this paper, both the full simulation and the proposed technique was executed for 2 minutes. The time spent for a single simulation step is recorded and presented in Figure 3. On average the proposed system is 10.8 times faster than the simulation without the levels of behaviour techniques. The system presented in this paper supports both flying and unpredicted camera movements. The level 1 locomotion helps the pedestians in unimportant places to be distributed logically. The distribution is verified by checking that the distribution of agents on the graph edges remains visually consistent with the full simulation. By widening the view frustum by 5 degrees in both directions and using the behaviour thresholds at the specified distances no collision artifacts are observed during transitions of the locomotion levels.

The maximum threshold values described in Section 3.1 are determined experimentally by the authors. However, different simulations may need different values and results from human perception experiments may be required in order to get the minimum possible threshold values for maximum efficiency.

## References


[CAD01] Chenney S., Arikan O., D.A.Forsyth: Proxy simulations for efficient dynamics. In *Eurographics 2001, Short Presentations.* (2001).

[MT97] Musse S. R., Thalmann D.: A model of human crowd behavior: Group interrelationship and collision detection analysis. In *Eurographics* (Budapest, Hungary, 1997), pp. 39–52.

[MYMT07] Morini F., Yersin B., Maïm J., Thalmann D.: Real-Time Scalable Motion Planning for Crowds. In *Cyberworlds* (2007).

[OCV*02] O'Sullivan C., Cassell J., Vilhjalmsson H., Dingaliana J., Dobbyn S., McNamee B., Peters C., Giang T.: Levels of detail for crowds and groups. *Computer Graphics Forum 21(4)* (2002).

[PdHCM*06] Pettré J., de Heras Ciechomski P., Maïm J., Yersin B., Laumond J.-P., Thalmann D.: Real-time navigating crowds: scalable simulation and rendering. *Comput. Animat. Virtual Worlds 17*, 3-4 (2006), 445–455.

[PGT08] Pettré J., Grillon H., Thalmann D.: Crowds of moving objects: navigation planning and simulation. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes* (New York, NY, USA, 2008), ACM, pp. 1–7.

[PLT05] Pettré J., Laumond J., Thalmann D.: A navigation graph for real-time crowd animation on multilayered and uneven terrain. In *V-CROWDS'05* (Lausanne, Switzerland, 2005).

[Rey99] Reynolds C. W.: Steering behaviors for autonomous characters. In *Game Developers Conference* (1999).