Intuitive Crowd Behaviour in Dense Urban Environments using Local Laws

Celine Loscos, David Marchal, Alexandre Meyer

▶ **To cite this version:**

Celine Loscos, David Marchal, Alexandre Meyer. Intuitive Crowd Behaviour in Dense Urban Environments using Local Laws. Theory and Practice of Computer Graphics, 2003, Birmingham United Kingdom. 10.1109/TPCG.2003.1206939 . hal-01498407

HAL Id: hal-01498407
https://hal.science/hal-01498407v1

Submitted on 30 Mar 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Intuitive Crowd Behaviour in Dense Urban Environments using Local Laws

Celine Loscos
University College London
C.Loscos@cs.ucl.ac.uk

David Marchal
Ecole Polytechnique Paris

Alexandre Meyer
University College London
A.Meyer@cs.ucl.ac.uk

Abstract

In games, entertainment, medical and architectural applications, the creation of populated virtual city environments has recently become widespread. In this paper we want to provide a technique that allows the simulation of up to 10,000 pedestrians walking in real-time. Simulation for such environments is difficult as a trade off needs to be found between realism and real-time simulation. This paper presents a pedestrian crowd simulation method aiming at improving the local and global reactions of the pedestrians. The method uses a subdivision of space into a 2D grid for pedestrian-to-pedestrian collision avoidance, while assigning goals to pedestrians to make their trajectories smoother and coherent. Goals are computed automatically and connected into a graph that reflects the structure of the city and triggers a spatial repartition of the density of pedestrians. In order to create realistic reactions when areas become crowded, local directions are stored and updated in real-time, allowing the apparition of pedestrians streams. Combining the different methods contributes to a more realistic model, while keeping a real-time frame rate for up to 10,000 simulated pedestrians.

1 Introduction

Pedestrian traffic may be quite congested in large cities. The design of most public facilities could often enable the passage of hundreds of thousands pedestrians per day. Railway stations, shopping centres or tourist places attract high density crowds. In order to help in the design or to demonstrate such places to a public, it is essential to be able to simulate the movement of a pedestrian crowd in such environments. Other applications such as games and entertainment should also benefit from such techniques.

Our primary goal is to provide a simulation that runs in real-time with no pre-computation. This simulation is part of a system that renders a city, simulates the crowd behaviour and renders each of the individuals. Resources are therefore shared, and the cost of the behaviour simulation must be restricted. This paper presents a way of simulating an intelligent crowd without using more complex approaches such as artificial intelligence techniques. Whereas most available models which describe the behaviour of a crowd usually deal with macroscopic variables like average and flow, we developed an individual-based model.

In the next section 2, we present related work dealing with virtual cities, real crowd features and former work on crowd modelling. In section 3 we describe how we define accessible regions by the pedestrians, using a 2D-analysis of the space. Then, in section 4, we explain the methods used to perform our main contributions: making the agents aware of the city environment, performing a realistic agent-to-agent collision avoidance, managing the self-emergence of flows and making the agents walk in small groups. In section 5, we present some details of implementation and the results.

2 Background

A good overview of human behaviour simulation in virtual environments is presented in [2]. In this section we will restrict the description of work to non-AI approaches. Moreover our main purpose is to give agents realistic trajectories at an individual level and to give the crowd movement a realistic look at a macroscopic level. There has been a lot of work intending to improve individual features of agents: in [1], for example, they add subconscious actions (such as walking stooped when sad) so that there may be a greater diversity of behaviours.

It is often interesting to base a simulation on real observations. Feurtey [4] collected a great number of crowd features, such as for example, the intimacy of the space needed, the relation between the flow of pedestrians, the density of the area and the speed, or the influence of the weather on the behaviour. These data can help to find what is significant in the way a crowd behaves so as to be able to incorporate this into a model.

Space syntax techniques [5] have a long tradition in urban studies. In the context of cities, space syntax aims at describing some areas in the sense of integration and segre-

gation. A location is more integrated if all the other places can be reached from it after going through a small number of intermediate places. With such parameters, the movement pattern in a city can be understood and predicted: for example, the more a region has extensive visibility from the surrounding area (with the longest lines of site), the busier it will be. Some space syntax analysis results even show correlations between predicted segregated areas and areas of high incidence of burglary.

Collision detection

In most simulations, attention is centered on collision detection and reaction. Bouvier and Cohen [3] implemented a simulation of a crowd involving 45,000 persons, based on Newtonian mechanics. However the simulated behaviour remains quite simple.

Reynolds [9] implemented, based on an exact mathematical computation, a large set of individual or group

behaviours such as pursuing a moving goal, obstacle avoidance, path following, or flow field following. The overall behaviour simulation combines these numerous kinds of individual behaviours. Although the results are really impressive, they are not scalable to a crowd of several thousands agents for real-time simulation.

Feurtey [4] proposed a new approach for collision detection based on predicting and modifying trajectories in a (x, y, t) space, using a cone to delimitate the available space based on the analysis of others' trajectories and speeds. Although this method allows pedestrians to evaluate the cost of moving away from their goal, changing direction, accelerating and decelerating, it is yet not scalable to a larger population.

Musse et al. [7] proposed a multi-resolution collision detection algorithm based on two different collision avoidance laws: the slower of two agents stops just before the collision occurs, or both agents go round each other. As the second method is much more expensive in computational time, it is used only when the observer position is really nearby the collision place. However this techniques has not been tested for a large number of pedestrians.

An original and efficient technique is proposed in [10], for the simulation of around 10,000 virtual humans. In this paper, the position of the pedestrians are managed with a 2D grid. They access, in the surrounding cells, local information for collision with the environment and other pedestrians. The idea has then been extended [11] for creating more complex behaviours using the same discretization approach, using different layers.

Group behaviour

In a city, less than a half of the pedestrians walk alone. Indeed most people walk in pairs. To simulate a realistic environment, it is necessary to implement group behaviour.

Using steering behaviours, Reynolds [9] invented the well-known *boids* (for bird-oid). Boids abide by a flocking rule that is simulated using the three *Separation*, *Cohesion* and *Alignment* basic steering behaviours. The result of the combination of these three laws was very good for the modelling of flocks, herds or schools but not collections of humans.

Musse et al. [6] defined a crowd as a set of groups formed by human agents, each group having a list of goals. Agents from a same group share the same list of goals but social effects can occur: agents may change group. To generate the group behaviour, they used 3 main laws: the agents from the same groups walk at the same speed, follow the same predefined path and can wait for each other when one agent is missing. In another paper, Musse et al. [7] used sociological rules to enable more human-like reactions. Each agent is specified by a level of dominance, a level of relationship and an emotional status, and is ruled by seeking and flocking laws. Using the same list of goals as before, the behaviour of the crowd is significantly improved: for example, they have implemented the simulation of 4 groups visiting a museum, each of the group aggregating gradually as time elapses. This is an example of global behaviour generated by local laws.

2.1 Discussion

Our work was inspired by techniques previously described, and have been adapted to local-control-based techniques used by Tecchia et al. [11]. We use a 2D-grid to localise the pedestrians, and for each of our contributions, we compute the decisions made by the pedestrians using information stored in the local surrounding cells. Although this seems to be very simplistic, it allows a high degree of control in the realism of the simulation with the development of complex rules, while keeping the computation cost low. In particular, our space analysis build on 2D maps to retrieve non-modelled information such as pavements and pedestrian crossings. Moreover we use several call-back layers proposed in [11] to manage flow streams, guidance in the directions of movement (goals) and group behaviour. We also used the crowd observations made by Feurtey [4] to assess the realistic appearance of our simulation. Finally, some collision detection algorithms developed by Musse et al. [6, 7] and Reynolds [9] contribute to our methods. Our collision detection algorithm is inspired by the previous observations [4, 6, 7] while implemented using simple local rules on the 2D discreet representation of the environment [11]. Architectural theories such as Space Syntax [5] link the physical aspects and the pedestrian business associated with a place. Although we do not use directly space syntax techniques, these make it possible to extend our algorithm using accurate laws, e.g. for taking into account of density.

3 Space Analysis

As it has been done before [10, 11, 13] and for efficiency reasons, we use a 2D array with a certain resolution to represent the physical city. The input data is a binary map representing only the position of the buildings and the ground. An example of such a map is shown in Fig. 1, where white regions represent the ground, and black regions the areas covered by the buildings. If pedestrians directly access this map, they can be given access to regions tagged by a GROUND value, and refused access to those tagged by a BUILDING value. Giving access to GROUND would only enforce the agents avoiding penetration of the buildings, but it would result in a very unrealistic behaviour. Streets are not only for pedestrians, and it would be more realistic to provide pavements and pedestrian crossings.

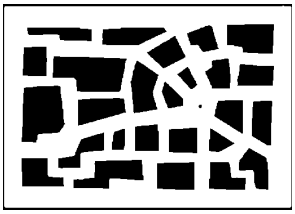


Figure 1. Binary image representing the virtual city. This can be automatically obtained by rendering the scene from above, using orthographic projection and storing the image in a z-buffer. A post-processing step might be needed to level the elevations (white for the ground and black for the buildings).

We used the 2D map given in input to redesign the area accessible by the pedestrians, by defining pedestrian pavements and crossings. Note that we did not model these components in 3D. In the following, we present the algorithm used, which is in three steps. First, we build the pavement areas directly from the 2D map. Second, we detect the corners of the pavements, and finally define pedestrian crossings. The method described in this section is the one we implemented, but we are aware that other techniques could be used. This one has the advantage of being simple and 2D-based.

3.1 Definition of the pavements

As our interest is to make the method work for any type of city, the input binary map representing the location of buildings is constructed by rendering the city from above using an orthographic projection. We render only the buildings without the polygons representing the road. In the resulting image, pixels with a depth value equal to infinity are the road (white) and others are the buildings (black). To define the pavement areas, we enlarge the area covered by the buildings using a convolution filter. The new pixels defined by the convolution are tagged as PAVEMENT.

3.2 Goal positioning

The following algorithm is invoked after having built pavements around buildings. To place the goals on the corners, we use a new more flexible method that needs to detect the succession of three different levels: GROUND-PAVEMENT-GROUND or PAVEMENT-BUILDING-PAVEMENT. This succession has to be found several times for the same corner with different angles (see Fig. 2) to validate the existence of a goal. This corner detection is performed on the pavements once we have built them with half of their final size. Afterwards, pavements are widened to their full width and the detected corners thus appeared to be in the centre of the pavements.

The inter-visible corners are formed into a graph. Although the number of goals is large, the method is conservative as there is no pavement area without a goal. This ensures, as later on described in section 4.1.1, that every pavement can be occupied by the pedestrians. The number of goals is then decreased by merging neighbour goals together when they are close, while keeping connectivity in the graph. If two goals A and B are to be merged, the resulting goal position is the barycentre of A and B and the new neighbours are those of A and B. This method is summarised in Fig. 3. It can be noticed that the property of the visibility between two connected goals in the graph might not be conserved after the merge. But this does not have a consequence for the rest of the algorithm.

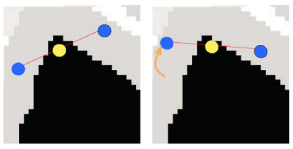


Figure 2. We identify a corner by detecting the sequence GROUND-PAVEMENT-GROUND or PAVEMENT-BUILDING-PAVEMENT. We show in this figure two of the tests on angle variation used to validate the existence of a corner.

3.3 Pedestrian crossings

Using the graph of goals, it is possible to automatically design the pedestrian crossings. For every pair of goals in the graph, it is possible to know if they are on opposite sides of a street. If it is the case, these two goals will be

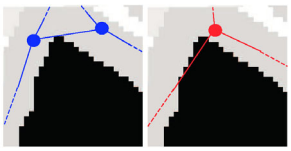


Figure 3. To decrease the number of goals, our algorithm merges goals. Two goals (left) are replaced by their barycentre (right).

Figure 4. This is an example of the automatic designing of pavements (in gray) and pedestrian crossings (in light gray).

set connected in the graph, and the corresponding 2D area covered tagged with a CROSSING value. Although it does not reflect exactly the way real pedestrian crossings lay, the pedestrian crossings created with this method correspond to a true necessity in terms of traffic (see Fig. 4).

4 Behaviour

In the previous section, we explained how we encoded the necessary information in a 2D grid to indicate to the pedestrians the accessible areas (CROSSING, PAVEMENT) and the non-accessible ones (BUILDING, GROUND). In section 4.1, we explain how this information is used to significantly improve the individual behaviour while keeping an interactive frame rate. We exploit the precomputed information to define trajectories and perform collision detection with the environment. We then explain how we enhanced the pedestrian behaviour to make it more realistic using simple local rules. In section 4.2, we describe a first attempt on simulating groups of people, still using 2D information.

4.1 Individual Behaviour

4.1.1 Trajectories

With our existing system, we were concerned with how to provide relatively straight trajectories to the pedestrians without adding too much complexity. The natural way is by giving virtual agents a direction. However, this results easily in a pedestrian moving toward an obstacle and then reacting to it. This easily makes the simulation similar to the behaviour of "ants" . A more sophisticated way is to give each agent a goals to attain, to retrieve a path to go from one point to another. This could be precomputed to avoid expensive computations at run time.

However we wanted a system that could be run without any precomputation, to encourage behavioural diversity and to reduce the memory storage need, while resulting in an interactive update of the behaviours. For this, we make use of the goals previously placed and used in the system (see section 3). At the begining of the simulation, every pedestrian has been placed next to a goal and has been assigned a new goal to reach, adjacent in the graph of goals, and thus visible from the current position. The direction from one goal to another corresponds roughly to the trajectory of a pedestrian on a pavement or a crossing. It is interesting to note that, therefore, frontal collision with static obstacles are unlikely to happen. When a pedestrian reaches a goal, a new goal is assigned from the list of the adjacent goals in the graph. To avoid virtual agents to do some u-turns, we stored into memory the last three goals assigned to prevent them to be assigned again.

4.1.2 Collision Detection

The pedestrians perform collision detection with the environment (buildings) and between themselves. As was previously proposed [10], we perform collision detection with the buildings using a collision map. In our simulation, we check up to five tiles ahead to avoid unpredicted collision, and therefore to contribute for the smoothness of the trajectory.

Detecting collision with people is also performed using the information stored in a grid. We consider three main cases of collision between two agents, *front*, *following* and *perpendicular*, as illustrated in Fig. 5. We compare the direction of trajectory of each agent, the velocity factor, and the distance between the agents. According to these parameters, a decision is taken, to deviate from an appropriate angle, to slow down, or to completely stop; the deviation being the preferred option. Depending on the distance to the other agent, the chosen variation can vary from $\pm\Pi/8$ when the distance is large to $\pm\Pi/2$ when the distance is short. The list of possible decisions is summarized in Fig. 7.

4.1.3 Flows

We observed that in everyday life, when a congestion occurs, people tend to follow the person in front of them. This is because we trust that the person in front of us probably made the right decision. We implemented this idea, using

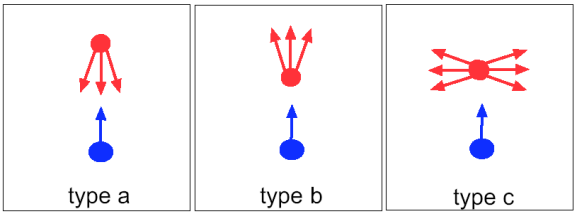


Figure 5. The three types of collision we consider. Type a: front collision. Type b: following collision. Type c: perpendicular collision.

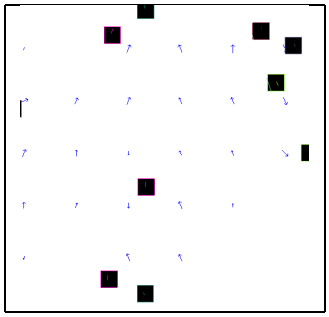


Figure 6. An example of the direction field used to simulate stream of people. Agents are represented by the squares, and the direction field is represented by the blue arrows. Notice that the size of the arrows varies as the validity of the field fades over time.

local information rather than space understanding or dynamic fluid simulation. We represent flows as a direction field, stored into the 2D grid. Every time an agent reaches a cell, its direction is stored. This direction field then fades over time to completely disappear. When an agent reaches a cell, it checks which direction was chosen by a previous agent and when. It then uses this information to make its own decision. The direction field is updated depending on this new direction, but taking into account for the previous direction stored. The equation used is

$$\overrightarrow{D(t)} = \left(\frac{t_0 - t}{\tau} + 1\right).\overrightarrow{D_0} \quad (1)$$

$$\overrightarrow{newD} = \frac{1}{2}\left(\left(\frac{t_0 - t}{\tau} + 1\right).\overrightarrow{D_0} + \overrightarrow{D_1}\right) \quad (2)$$

where $D(t)$ is the way the stored direction fades over time, the maximum time interval being τ, and t_0 being the time when an agent updated D. Using equation (2), a new direction $newD$ is computed, where D_0 is the previously stored direction and D_1 the original direction of the new agent. An example of a direction field is shown in Fig. 6. It is interesting to notice that agents in this context cooperate. A decision taken by an agent in a cell influences the next occupier.

4.1.4 Adaptation to density

In our everyday life, depending on the density of a crowd, we usually adopt different behaviours. We wanted to take this into account in our simulation by encouraging pedestrians to queue in busy areas and to adapt their speed instead of overtaking. Instead of analysing the density at each step of the simulation, we use a density prediction map. In our case we noticed that the density is highest in the region where we placed more goals. This is coherent since it is more likely that a pedestrian is assigned a goal from a high-density goal area. However, this has nothing to do with a space analysis and it would be more interesting to use a density prediction map as produced by space syntax techniques. If we were to use these maps, we could then either simply assign more goals in the area where higher intensity is predicted, or weight the choice of the goals by the space syntax density encoded information.

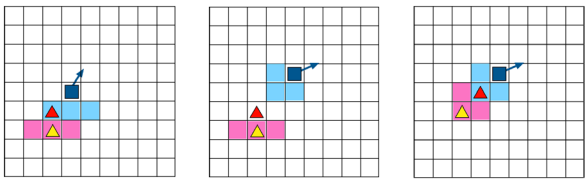
In our simulation, the density factor influences decisions when pedestrians are to collide with one another. In a congested area, the far range is ignored (we do not check ahead as cells are often closely occupied). In Fig. 7, we summarize the decision taken by an agent when a collision is predicted. When the distance is close, it means that the target cell is occupied. When the distance is near, it means that an agent occupies a cell either 2 or 3 cells away. If the distance is far, it means that an agent occupies a cell either 4 or 5 cells away.

4.2 Group Behaviour

The issue of making people walk in small groups has two main goals: the first one is obviously to improve the reality of the simulation. From observations in the street, one can conclude that around half of the pedestrians walk in pairs or more. Moreover, using groups could be a mean to reduce the computing time for the same number of agents. The definition of a group [7] is often given by a set of people who have the same goals or list of goals and the same emotional parameters, that means, in our case, the same average speed and way of accelerating. We define a group by a *leader* and *members*. The *leader* takes the decision relative to the direction to take, and the *members* follow. *Members* still perform collision detection before moving using the laws described in section 4.1, but their choice is influenced by the *leader*. Providing the number of operations necessary to compute the *members*' behaviour is smaller than the *leader*'s , the average frame rate of the simulation is likely to increase.

Distance	Behaviour of the other	Type of collision	Reaction in normal conditions	Reaction in traffic congestion
Close	Waiting	front	If possible, overtake with angle $\pm\Pi/2$, else slow down or wait	If possible, overtake with angle $\pm\Pi/2$, else slow down or wait
Close	Waiting	following or perpendicular	If possible, overtake with angle $\pm\Pi/2$, else slow down or wait	Wait
Close	Walking	front	If possible, overtake with angle $\pm\Pi/2$, else slow down or wait	If possible, overtake with angle $\pm\Pi/2$, else slow down or wait
Close	Walking	following	Take the same linear speed as the other	Take the same linear speed as the other
Close	Walking	perpendicular	Wait	Wait
Near	Waiting	front	If possible, overtake with angle $\pm\Pi/4$, else slow down or wait	If possible, overtake with angle $\pm\Pi/4$, else slow down or wait
Near	Waiting	following	Slow down	Slow down
Near	Waiting	perpendicular	If possible, overtake with angle $\pm\Pi/4$, else slow down or wait	Slow down
Near	Walking	front	If possible, overtake with angle $\pm\Pi/4$, else slow down or wait	If possible, overtake with angle $\pm\Pi/4$, else slow down or wait
Near	Walking	following	Take the means of both linear speeds	Take the same linear speed as the other
Near	Walking	perpendicular	Slow down or wait	Slow down or wait
Far	Waiting	following	Slow down	
Far	Waiting	front or perpendicular	If possible, overtake with angle $\pm\Pi/8$, else slow down or wait	
Far	Walking	front	If possible, overtake with angle $\pm\Pi/8$, else slow down or wait	
Far	Walking	following	No special reaction	
Far	Walking	perpendicular	Slow down	

Figure 7. Decision taken by a pedestrian to prevent collision with another pedestrian, depending on the density, the distance, the directions, and the current behaviour of the other one.



(a) (b) (c)

Figure 8. Group behaviour method: the square is the *leader* and the two triangles are *members* (triangles). (a) The *leader* advised 3 tiles (light blue) and so did the first member (light red). (b) The *leader* has moved to a new cell and advised new tiles. (c) The two *members* have moved to keep the connectivity of the group.

In real situations, the size of pedestrian groups is rarely bigger than 3. However, tourist groups can be composed of around 20 people. These differences though are often levelled due to the density of the traffic. It is quite rare to see ten people aligned; the group is more likely to split into smaller sub-groups of 2 or 3 pedestrians, each sub-group following the other. In our simulation, we decided to initialise the number of groups randomly using the probability law: 95% of the groups have a size of 1 or 2; 5% have a size between 3 and 10.

The group moves in a sequential way: the *leader* of a group always moves first. Once it has moved, it tags 3 tiles behind its position with an *advised tile* flag. Then, the *members* of the group move one by one in the same frame: they have to move to an *advised tile*. In order to have enough *advised tiles* for the whole group, every member, after moving, advises the 3 tiles behind it with the tag. When every member has moved, all the *advised tile* tags are deleted. Fig. 8 illustrates this method.

We noticed that this method, as we currently implemented it, suffers from some limitations. First, the *leader*, taken as a pedestrian, is always ahead of the others. We thus decided not to display it while continuing using it for the lead. Second, disconnections between *members* of the group occur when advised tiles are not well connected, or when *members* could not reach an advised tile. We are still working on this method to improve it.

5 Implementation and Results

We implemented the algorithms described in the previous sections. The architecture of the program is described in Fig. 9. Before starting the simulation, each agent is assigned a goal, a position, a direction, a speed vector, and a group. During the simulation, they access the cells corresponding of their current position. However, they have a discreet number of positions within each cell to make the motion smoother. This can however produces some visual inaccuracy in the collision detection algorithm, when two pedestrians occupy two different cells, but have a very nearby position.

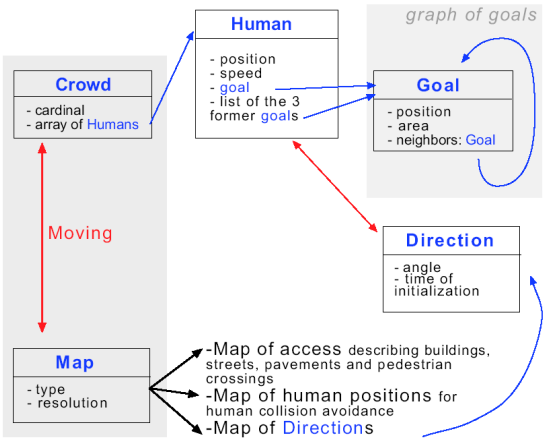
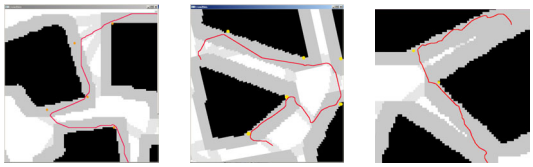


Figure 9. The architecture of the behaviour platform.

We tested the behaviour of the agents by tracking them in the environment. When pedestrians are not in the field of view of the camera, their behaviour is updated, but they do not perform collision detection. In Fig. 10, we tracked one agent in different conditions. The trajectory chosen by the agent is satisfactory since it is coherent and smooth.

We imported the behaviour platform into our rendering system [12, 13, 8] and the results are shown in Fig. 11. Our simulation runs on a 2Ghz Intel Pentium PC with a GeForce4 Ti4600 graphics card. We measured the average frame rate when simulating with a different number of pedestrians, as shown in the graph in Fig. 12.



(a) (b) (c)

Figure 10. On (a) and (b), we can see the trajectory followed by an agent without being seen by the camera (that is without agent-to-agent collision detection performed) during simulation involving respectively 1,000 and 10,000 agents. On (c), an agent has been tracked (with the agent-to agent collision detection activated) in a busy street; the simulation runs with 5,000 agents. We can see that the trajectory is less smooth than in (a) and (b) due to collision avoidance with other agents.

In its current implementation, our algorithm is not sufficient for real-time (more than 25 frame per seconds) when we simulate more than 6,000 people. However, the frame rate is reasonable even for 10,000 as the frame rate stays above 10. Additional results can be seen in http://www.cs.ucl.ac.uk/research/vr/Projects/Crowds/EGUK03/.

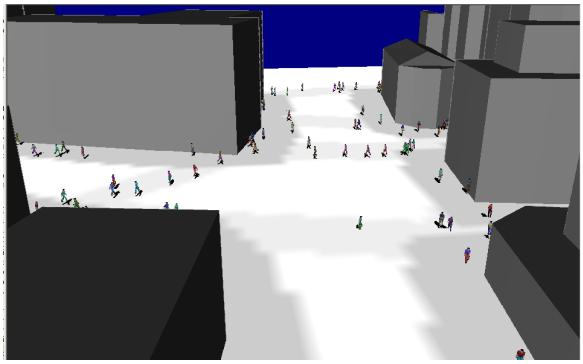


Figure 11. Example of simulation of our work integrated in the rendering system. The simulation runs with 7,000 pedestrians.

6 Conclusion and future work

We presented a new technique to improve local behaviour of virtual pedestrians moving in a city while managing the complexity and keeping a real-time frame rate.

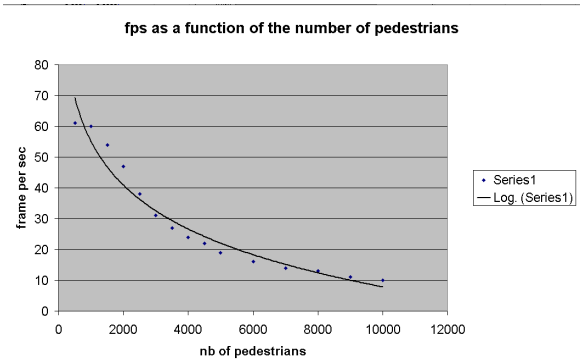


Figure 12. The number of frame per second decreases less and less with the number of pedestrians. For up to 6,000 agents, the animation remains fluid.

Our method uses a 2D discretization of the space to perform local decisions. We presented automatic techniques to detect areas accessible by pedestrians. Pedestrians are able to control their trajectory by storing the aimed direction and performing collision detection with the environment. Inter-collision detection is also performed and decision are made depending on parameters such as speed, direction, and density. We simulate two kinds of behaviour, one in normal traffic conditions, the other one when congestion occurs. In high-density regions, we adapt the inter-collision detection algorithm and we provide cooperation between agents for making decision. Finally, we implemented an algorithm to simulate group behaviour.

Obviously, even if the behaviour has been greatly improved while keeping a low frame rate, further complex behaviours need to be simulated. Agents should be able to enter buildings, stop at bus stops, slow down to do some window shopping, or walk in pairs as parent and child. More complex animation should be explored, allowing pedestrians to seat on benches, wait, or interact together. Better integration with space syntax techniques could allow more accurate occupation of the space. Finally we believe that we could exploit ideas from this paper to provide a simulation for vehicles in a city.

Acknowledgement

The research described in this paper has been funded with an EPSRC project, Equator, and with an IST EU project, CREATE (IST-2001-34231).

References

[1] Koji Ashida, Seung-Joo Lee, Jan M. Allbeck, Harold Sun, Norman I. Balder, Dimitris Metaxas, Pedestrians: creating agent behaviors through statistical analysis of observation data, Computer Animation, 2001.

[2] Ruth Aylett, Marc Cavazza: Intelligent Virtual Environments- A state-of-the-art report, Eurographics 2001 - STARs, 2001.

[3] Bouvier, Cohen, simulation of human flow with particles systems, Proceedings of the 1995 Simulation MultiConference, 1994.

[4] Franck Feurtey, Takashi Chikayama, Simulating the collision avoidance of pedestrians, Masters thesis, University of Tokyo, February 2000.

[5] Bin Jiang, Christophe Claramunt, Björn Klarqvist, An integration of space syntax into GIS for modelling urban spaces, *JAG*, Volume 2, Issue 3/4, 2000.

[6] Soraia R. Musse, Christian Babski, Tolga Capin and Daniel Thalmann, Crowd Modelling in Collaborative Virtual Environments, *ACM VRST'98*, Taiwan,1998.

[7] S.R. Musse, D. Thalmann, a model of human crowd behavior: group inter-relationship and collision detection analysis, *Proc Workshop of Computer Animation and Simulation of Eurographics'97*, Sept 1997, Budapest, Hungary.

[8] Céline Loscos, Franco Tecchia, Yiorgos Chrysanthou: Real Time Shadows for Animated Crowds in Virtual Cities, *ACM Symposium on Virtual Reality Software & Technology 2001 (VRST01)*, Banff, Alberta, Canada, November 2001.

[9] C.W. Reynolds, Steering behaviours for autonomous characters, *Game Developers Conference*, Miller Freeman Game Group,1999.

[10] Franco Tecchia, Yiorgos Chrysanthou: Real-time visualisation of densily populated urban Environments: a simple and fast algorithm for collision detection, *Eurographics UK*, April 2000.

[11] Franco Tecchia, Céline Loscos, Ruth Conroy, Yiorgos Chrysanthou: Agent behavior simulator (ABS): a platform for urban behavior development, *Proc. Game Technology* (GTEC 2001),CD-ROM, 2001.

[12] Franco Tecchia, Céline Loscos, Yiorgos Chrysanthou: Image-based crowd rendering, *IEEE computer graphics and applications*, March/April 2002, p. 36-43.

[13] F. Tecchia and Y. Chrysanthou: Real-Time Rendering of Densely Populated Urban Environments, *Eurographics Rendering Workshop 2000*, p. 83-88, Springer Computer Science, 2000, Rendering Techniques 2000.