Vis Comput (2008) 24: 859–870
DOI 10.1007/s00371-008-0286-0

ORIGINAL ARTICLE

Real-time crowd motion planning

Scalable Avoidance and Group Behavior

Barbara Yersin · Jonathan Maïm · Fiorenzo Morini · Daniel Thalmann

Published online: 7 August 2008

Abstract Real-time crowd motion planning requires fast, realistic methods for path planning as well as obstacle avoidance. In a previous work (Morini et al. in Cyberworlds International Conference, pp. 144–151, 2007), we introduced a hybrid architecture to handle real-time motion planning of thousands of pedestrians. In this article, we present an extended version of our architecture, introducing two new features: an improved short-term collision avoidance algorithm, and simple efficient group behavior for crowds. Our approach allows the use of several motion planning algorithms of different precision for regions of varied interest. Pedestrian motion continuity is ensured when switching between such algorithms. To assess our architecture, several performance tests have been conducted, as well as a subjective test demonstrating the impact of using groups. Our results show that the architecture can plan motion in real time for several thousands of characters.

Keywords Crowds · Real-time · Motion planning · Groups

1 Introduction

Realistic real-time motion planning for crowds has become a fundamental research field in the Computer Graphics community. The simulation of urban scenes, epic battles, or other environments that show thousands of people in real time require fast and realistic crowd motion. Domains of application are vast: video games, psychological studies, and architecture, to name a few. In this paper, we present an improved architecture offering a hybrid, scalable solution for real-time motion planning of thousands of characters in complex environments. Moreover, to improve crowd behavior, a simple and efficient approach to simulate small groups of pedestrians is introduced.

In our perspective, crowds are formed by thousands of individuals that move in a bounded environment. Each pedestrian wants to reach his individual goal in space, avoiding obstacles, and remaining close to his friends or family. People perceive their environment, and use this information to choose the shortest path in time and space that leads to their goal. Emergent behaviors can also be observed in crowds, e.g., in places where the space is small and very crowded, people form lanes to maximize their speed. Also, when dangerous events occur, pedestrians tend to react in very chaotic ways to escape.

Planning crowd motion in real time is a very expensive task, which can be divided into three distinct parts: path planning, obstacle avoidance, and group cohesion. Path planning consists in finding the best way to reach a goal. The path selection criteria are the avoidance of congested zones, and minimization of distance and travel time. Path planning must also offer a variety of paths to spread pedestrians in the whole scene. Obstacles to avoid can either be other pedestrians or objects that compose the environment. The goal of each individual is to inhibit collisions with such obstacles. As for groups, they are very often represented by 2 to 4 pedestrians walking side by side, and avoiding separation as well as inter-collision. In the context of real-time

B. Yersin (✉) · J. Maïm · F. Morini · D. Thalmann
IC ISIM VRLAB, Station 14, 1015 Lausanne, Switzerland
e-mail: barbara.yersin@epfl.ch

J. Maïm
e-mail: jonathan.maim@epfl.ch

F. Morini
e-mail: fiorenzo.morini@gmail.com

D. Thalmann
e-mail: daniel.thalmann@epfl.ch

simulations, all three aspects of motion planning need to be addressed efficiently to produce believable results.

Multiple motion planning approaches for crowds have been introduced. As of today, several fast path planning solutions exist. Dynamic avoidance and high-level group behaviors, however, remain expensive tasks. Agent-based methods offer realistic pedestrian motion planning, especially when coupled with global navigation. This approach gives the possibility to add individual and cognitive behaviors to each agent, but becomes too expensive for large crowds. Potential field approaches handle long- and short-term avoidance. Long-term avoidance predicts possible collisions and inhibits them. Short-term avoidance intervenes when long-term avoidance cannot prevent a collision. These methods offer less believable results than agent-based approaches, because they do not provide the possibility to individualize each pedestrian behavior. However, they have much lower computational costs.

In this paper, we detail our improved architecture that realistically handles crowd motion planning in real time. Our approach provides a complete solution for all three aspects of crowd motion, i.e., path planning, obstacle avoidance, and group behavior. To obtain high performance, our approach is scalable: we divide the scene into multiple regions of varying interest, defined at initialization and modifiable at runtime. According to its level of interest, each region is ruled by a different motion planning algorithm. Zones attracting the user attention exploit accurate methods, while computation time is saved with less expensive algorithms in other regions. Our architecture also ensures that no visible disturbance is generated when switching from one algorithm to another.

Results show that we can simulate up to 10,000 pedestrians in real time with a large variety of goals. Also, small groups are created and their members keep close to each other, as in reality. Finally, the possibility to introduce and interactively modify the regions of interest in a scene allows the user to choose the performance and distribute computation time accordingly. We illustrate in Fig. 1 pedestrians taking advantage of our architecture to plan their motion in two environments.

Overview In this paper, we first introduce related work in Sect. 2. Then, in Sect. 3, we describe our architecture, and how we exploit it to distribute regions of three different levels of interest. In Sect. 4, the integration of the various approaches employed and the optimizations applied to keep high frame rates are detailed. Section 5 is dedicated to our high-level group behavior algorithm. In Sect. 6, we show our results in terms of performance as well as subjective individual ratings in different conditions and environments to assess our architecture. Finally, limitations and future work are discussed in Sect. 7.

2 Related work

Crowd behavior and motion planning are two topics that have long been studied in fields such as Robotics and Sociology. More recently, however, and due to the technological improvements, these domains have aroused the interest of the Computer Graphics community as well.

The first studied approach, i.e., agent-based, represents a natural way to simulate crowds as independent individuals interacting with each other. Such algorithms usually handle short distance avoidance, in which navigation remains local. Reynolds [24] proposed to use simple rules to model crowds and groups of interacting agents. Musse and Thalmann [17] used sociological concepts in order to simulate relationships among virtual humans. Niederberger and Gross [18] introduced a generic system for autonomous and reactive agents, organized in hierarchical groups. Shao and Terzopoulos [26] used perceptual, behavioral, and cognitive models to simulate individuals. Heïgeas et al. [4] introduced a model based on cellular automata and the physical properties of the environment, while Kirchner and Shadschneider [10] used static potential fields to rule a cellular automaton. Metoyer and Hodgins [15] proposed an avoidance algorithm based on a Bayesian decision process. Nevertheless, the main problem with agent-based algorithms is their low performance. With these methods, simulating thousands of pedestrians in real time requires the use of particular machines supporting heavy parallelizations [25]. Moreover, such approaches

Fig. 1 Pedestrians using our hybrid motion planning architecture to reach their goal, avoid each other, and form small groups (*Left*) in a large landscape of fields, and (*Right*) in a city environment

forbid the construction of autonomous adaptable behaviors, and can only manage crowds of pedestrians with local objectives.

To solve the problems inherent in local navigation, some behavioral approaches have been extended with global navigation. Bayazit et al. [1] stored global information in nodes of a probabilistic roadmap to handle navigation, and introduced simple rules in order to manage groups. Sung et al. [27] first introduced an architecture for developing situation-based crowd behavior, including groups. They later combined probabilistic roadmaps with motion graphs to find paths and animations to steer characters to a goal [28]. Lau and Kuffner [12] used precomputed search trees of motion clips to accelerate the search for the best paths and motion sequences to reach an objective. Lamarche and Donikian [11] used automatic topological model extraction of the environment for individual navigation. Another method, introduced by Kamphuis and Overmars [9], allowed a group of agents to maintain a given cohesion while trying to reach a goal. Although these approaches offer appealing results, they are not fast enough to simulate thousands of pedestrians in real time. Loscos et al. [14] presented a behavioral model based on a 2D map of the environment. Their method is suited for simulating wandering crowds, but does not provide high level control on pedestrian goals. Pettré et al. [21, 22] presented a novel approach to automatically extract a topology from a scene geometry and handle path planning using a *navigation graph* (see Fig. 2 (left)). The main advantage of this technique is that it handles uneven and multi-layered terrains. Nevertheless, it does not treat inter-pedestrian collision avoidance. Finally, Helbing et al. [5, 6] used agent-based approaches to handle motion planning, but mainly focused on emergent crowd behaviors, in particular scenarios.

Another approach for motion planning is inspired from fluid dynamics. Such techniques use a grid to discretize the environment into cells. Hughes [7, 8] interpreted crowds as density fields to rule the motion planning of pedestrians. The resulting potential fields are dynamic, guiding pedestrians to their objective, while avoiding obstacles. Chenney [2] developed a model of flow tiles that ensures, under reasonable conditions, that agents do not require any form of collision detection at the expense of precluding any interaction between them. More recently, Treuille et al. [29] proposed realistic motion planning for crowds. Their method produces a potential field that provides, for each pedestrian, the next suitable position in space (a *waypoint*) to avoid all obstacles.

Compared to agent-based approaches, these techniques allow to simulate thousands of pedestrians in real time, and are also able to show emergent behaviors. However, they produce less believable results, because they require assumptions that prevent treating each pedestrian with individual characteristics. For instance, only a limited number of goals can be defined and assigned to sets of pedestrians. The resulting performance depends on the size of the grid cells and the number of sets.

More recently, techniques using real captured data have emerged [13, 19]. Also, hybrid approaches have been proposed. Pelechano et al. [20] combined psychological, physiological, and geometrical rules with physical forces to simulate dense crowds of autonomous agents. In a recent work, Morini et al. [16] introduced a hybrid architecture offering a scalable solution for real-time crowd motion planning. Based on a navigation graph, the environment is divided into regions of varying interest. In regions of high interest, a potential field-based approach is exploited. Since potential fields are used only locally, motion is planned for many more sets and with finer grid cells than with an algorithm purely based on potential fields. In other regions, motion planning is ruled by a navigation graph and a short-term collision avoidance algorithm.

In this paper, we fully detail this hybrid architecture and its implementation. Moreover, significant improvements are introduced: an efficient and more realistic short-term avoidance algorithm as well as a new and simple approach to introduce small group behaviors within a moving crowd.

3 Architecture

The foundation of our architecture is based on navigation graphs, automatically extracted from the mesh of an arbitrary environment [21]. This approach has the advantage of

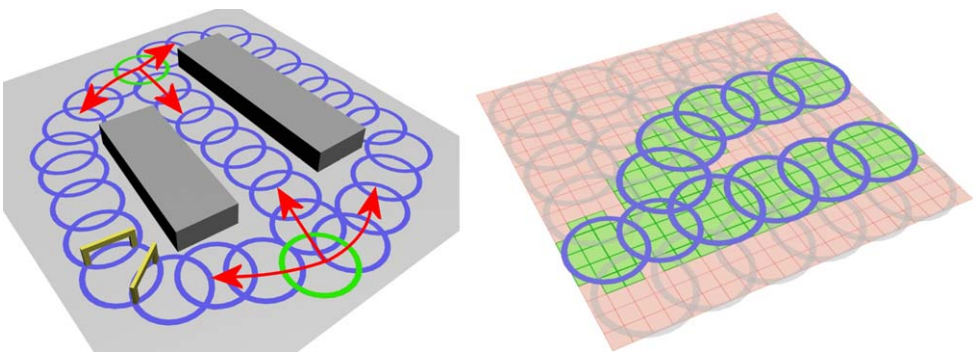


Fig. 2 (*Left*) A navigation graph composed of a single navigation flow (in *blue*) connecting two distant vertices (in *green*). The navigation flow is composed of three paths that can be followed in either direction (*red arrows*). Two edges are also represented as gates (in *yellow*). (*Right*) Grid placed on top of the graph. Only cells within a vertex that is part of a path stay active (in *green*)

robustly handling path planning. Vertices represent cylindrical zones of the walkable space, while edges are the gates where pedestrians can cross the space from one vertex to another. To connect two distant vertices, it is possible to create a *navigation flow*, composed of a set of varied paths. An example is shown in Fig. 2 (left). Thanks to this approach, pedestrian spreading is ensured. During simulation, pedestrians are assigned one navigation flow, and one direction. When they reach an extremity of the flow, they reverse their direction, and choose a new path, minimizing their travel time, e.g., avoiding congested areas. Vertices offer a suitable structure of the walkable space; they can be exploited to classify different regions of the scene. For instance, Pettré et al. [21, 22] used them to define several levels of simulation, each updated at different frequencies.

The goal of our architecture is to handle thousands of pedestrians in real time. We thus exploit the above-mentioned vertex structure to divide the environment into regions ruled by different motion planning techniques. We classify these regions with a level of interest. The most interesting zones are ruled by realistic but expensive techniques, while others use simpler and faster solutions. Regions of interest (ROI) can be defined in any number and anywhere in the walkable space with high-level parameters, modifiable at runtime. Such flexibility is indeed desirable: it allows the user to first choose the wanted performance, and then distribute ROI, i.e., computation time, as wished.

By defining three different ROI, we obtain a simple and flexible architecture for realistic results: **ROI 0** is composed of vertices of *high* interest, **ROI 1** regroups vertices of *low* interest, and **ROI 2** contains all other vertices, of *no* interest. With this classification, it is possible to divide the environment into many zones, each tagged with the appropriate level. In practice, we position the ROI with respect to the camera position and field of view. ROI 0 is directly in front of the camera, and in zones where important events occur. ROI 1 covers the remaining visible space, while ROI 2 includes all vertices outside the view frustum. Note that this choice is arbitrary, and that our architecture is versatile enough to satisfy any other environment decomposition.

For regions of no interest (ROI 2), path planning is ruled by the navigation graph. Pedestrians are linearly steered to the list of waypoints on their path edges. To use the minimal computation resources, obstacle avoidance is not handled.

Path planning in regions of low interest (ROI 1) is also ruled by the navigation graph. To steer pedestrians to their waypoints, an approach similar to Reynolds' is used [24]. Obstacles are avoided, thanks to a new agent-based short-term algorithm (see Sect. 4.4), providing efficient and more realistic results than our previous approach [16].

In the regions of high interest (ROI 0), path planning and obstacle avoidance are both ruled by a potential field-based algorithm, similarly to Treuille et al. [29]. Compared to agent-based approaches, potential fields are less expensive, and still offer results more realistic than the ones of ROI 1 and ROI 2, because collision avoidance is planned in the long term. Nevertheless, in certain situations, this approach fails to avoid collisions. To overcome this problem, the same short-term algorithm as in ROI 1 is also activated in ROI 0.

Group behavior is an additional layer of the architecture that can be used in order to simulate pedestrians walking with friends or family. This algorithm, detailed in Sect. 5 is purely based on the edition of pedestrian waypoints. Thus, it can be used for all ROI introduced above.

An important concern when dealing with regions ruled by different motion planning algorithms is to keep smooth and unnoticeable transitions at their borders. The way we place ROI implicitly solves this issue. Firstly, ROI 2 is always outside the view frustum, and thus does not require any specific attention. Secondly, passing the borders between ROI 0 and ROI 1 is always smooth, because they both use the same short-term avoidance algorithm.

4 Implementation

In this section, we present the details of our hybrid architecture implementation. We mainly focus on the initialization and runtime operations to construct and manage the scalable crowd motion planning. Firstly, in Sect. 4.1, the initialization phase is detailed, i.e., the grid construction over the graph space, the initialization of the structure of neighbor cells and of the ROI. Then, we describe our runtime pipeline, composed of five stages: the classification of graph vertices in correct ROI (Sect. 4.2), the potential field computation (Sect. 4.3), our short-term avoidance computation algorithm (Sect. 4.4), the pedestrian steering phase (Sect. 4.5), and finally, the continuity maintenance between grid and navigation graph (Sect. 4.6). Generating and handling small moving groups also require specific operations at initialization and runtime. We cover these steps in Sect. 5.

4.1 Initialization

First of all, for the given environment, a navigation graph is generated, and navigation flows created. We maintain a list of all active vertices, i.e., vertices belonging to at least one path. The others are discarded, since no pedestrian will ever pass through them during simulation. Then, a grid is disposed on the scene, its size limited by the bounding rectangle containing all graph vertices. This grid is composed of an array of cells, each containing the link to its neighbor cells, and intrinsic parameters used to compute the potential.

Many of the cells that compose the grid are not needed in the simulation, because they represent zones that are not

covered by graph vertices, and thus indicate static obstacles. Moreover, some vertices are not used by any navigation flow, and are not exploited by pedestrians, as illustrated in Fig. 2 (right). Thus, for each cell, we test whether its center is inside a vertex that composes a path; if not, the cell is deactivated. The main advantage of this preprocess is the reduction of the number of cells in which the potential field computation is necessary. Finally, each cell is linked to its active neighbors only.

4.2 Classification of vertices in ROI

To define a ROI, the user specifies three parameters: a position, a radius, and a level of interest. All vertices whose center is contained within this region are assigned the specified level. These parameters can be modified at any moment, implying a re-classification of vertices.

In our practical use of ROI, we create three lists corresponding to our three levels of interest. At runtime, we first automatically detect vertices that are outside the view frustum, and put them into the list with the lowest level of interest (ROI 2). We then iterate over the remaining vertices, testing whether they are inside ROI 0. If it is the case, the vertex is classified as of high interest and put in the corresponding list. Otherwise, it is put in the remaining list, of low interest (ROI 1). In the next two sections, we detail how pedestrian motions are planned in ROI 0 and ROI 1.

4.3 Potential field computation

To accelerate the potential field computation, it is possible to group pedestrians, as suggested by Treuille et al. [29]. It is important here to note that the term "group" can be confusing when used on the one hand for potential field computation, and on the other hand, to describe grouping behaviors within the crowd. In order to avoid ambiguity in the remaining of the paper, the term "set" is used to describe pedestrians with the same goal. We refer to "groups" as small teams of pedestrians walking together. For the set creation, pedestrians in ROI 0 sharing the same navigation flow and direction, i.e., having the same goal, are brought together in one set. Thus, there are two sets for each of these navigation flows. Sets are recomputed at each time step, to correctly classify pedestrians that change ROI.

Once the sets are created, a potential field is computed for each of them. At the goal, the potential is set to 0, and increased over the grid. Given the potential gradient, each pedestrian is assigned a new waypoint, corresponding to the center of a neighbor cell. For further details on the potential field computation, see [29]. Taking advantage of our architecture, we introduce a technique to reduce the computation time. Actually, the potential field is only required in regions of high interest (ROI 0). These regions cover part of the scene, and thus part of the grid. By computing the potential only for the cells located inside ROI 0, we can drastically decrease computation time. However, goals are often outside these regions, and thus, it is impossible to initiate the potential computation. For each set, we therefore create subgoals, situated just outside ROI 0, as shown in Fig. 3. We use the navigation flow structure to identify them: for every path of every flow leaving ROI 0, the first vertex met in the direction of the goal is a subgoal. The potential computation is initiated in the central cell of every subgoal, and spread over all cells inside ROI 0. To obtain the same behavior as if the potential was computed all over the grid, we do not initiate the potential of the subgoal cells to 0, but approximate it. For each subgoal cell c inside subgoal vertex v_c, the potential ϕ_c is computed as:

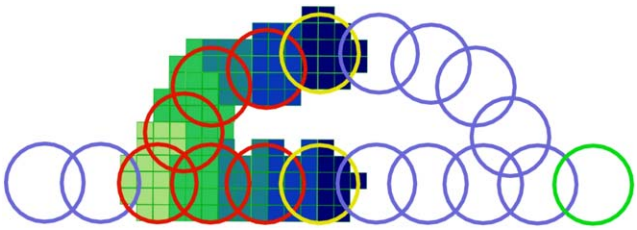


Fig. 3 Potential is computed for vertices either in ROI 0 (in *red*) or identified as subgoals (in *yellow*). The final goal is displayed in *green*. Potential starts in the central cells of the subgoals with an approximated value

$$\phi_c = C \cdot \sum_{v \in P(v_c)} (v.density + 1) \cdot v.radius \tag{1}$$

where v is a vertex of path $P(v_c)$, starting at v_c and leading to the final goal. The density of v is given by the number of pedestrians in it per square meter. Thus, the contribution to the potential of each vertex v is defined as its radius, weighted by its degree of occupation. To avoid a null contribution from an empty vertex, we always add 1 to the computed density. Constant C is used to weight the sum so that values for ϕ_c are in the same range as if the potential was computed from the goal. Note that vertex v_c may be part of several paths at the same time. In this case, we compute (1) for each path, and assign the lowest result to ϕ_c.

4.4 Improved short-term avoidance algorithm

In this section, we introduce our improved short-term avoidance algorithm, based on the assumptions that pedestrians mostly want to first maximize their speed and then to minimize detours.

This algorithm employs the grid cells to efficiently avoid local inter-pedestrian collisions in both ROI 0 and ROI 1. Particularly, in ROI 0, it complements the potential field approach, which may fail when the available space is too small and too crowded.

Note that the presented method is generic and adjustable to fit special requirements. For instance, if no implementation of a potential field approach is available, this algorithm could be adapted to predict possible collisions up to several meters ahead. Thus, it can perfectly be used as an intermediate or replacement solution for long-term and short-term avoidance methods. Its implementation is summarized in Algorithm 1 and further detailed below.

To find pedestrians that can potentially collide, we take advantage of the grid structure covering the whole environment: at runtime, every pedestrian in ROI 0 or ROI 1 is registered in its current grid cell, as shown in Algorithm 1, line 3. In this way, we can reduce the search for possible collisions to a small set of neighbor cells. Although this simplification does not cut down the order of complexity in $O(n^2)$, it significantly decreases n, as compared to a brute force approach [23].

Once pedestrians are registered in their current cell, we proceed in three important steps. Firstly, we identify cells ahead of the pedestrian, depending on its speed and direction. Secondly, a security check is performed for pedestrians in a certain range of cells. Finally, an emergency avoidance is performed if the pedestrians are closer than an emergency threshold β.

Find cells ahead For each pedestrian p in ROI 0 or ROI 1, we identify cells in its forward direction. This is achieved by defining three different vectors (line 7): f_p directly in front of p, l_p to its left, and r_p to its right. The front vector is filled with cells as follows:

$$f_p[i] = getCell(c.pos + i \cdot c.w \cdot p.fwd) \tag{2}$$

where $c.pos$ is the center position of cell c in which the current pedestrian p is situated. The normalized vector $p.fwd$ represents p's current forward direction, and $c.w$ is the width of a generic cell. Similarly, the vectors l_p and r_p are filled:

$$l_p[i] = getCell(c.pos + l \cdot c.w + i \cdot c.w \cdot p.fwd), \tag{3}$$

$$r_p[i] = getCell(c.pos - l \cdot c.w + i \cdot c.w \cdot p.fwd), \tag{4}$$

where l is the normalized vector perpendicular to $p.fwd$ on its left-hand side. The number of cells used to fill the vectors is arbitrary. In our implementation, we opted for vectors of variable size, depending on the pedestrian's speed. Thus, a slow pedestrian looks less far ahead than a fast one. These filled vectors form a corridor always pointing in the forward direction of the pedestrian. An example is illustrated in Fig. 4, where the identified cells ahead of the pedestrian are highlighted. Note that inactive cells are not taken into account when filling the vectors.

Algorithm 1 Improved short-term avoidance algorithm

```
   Data: set of pedestrians in {ROI 0 ∪ ROI 1}, set of
         grid cells, distance β
   Result: updated set of pedestrians in
           {ROI 0 ∪ ROI 1}
 1 if isEven(frameNumber) then
 2     for each pedestrian p ∈ {ROI 0 ∪ ROI 1} do
 3         register p in its current cell c_p
 4 else
 5     // emergency check
 6     for each pedestrian p in cell c_p do
 7         find n cells ahead to fill f_p, l_p, r_p.
 8         for each pedestrian p_neighbor in {f_p, l_p, r_p}[i]
 9         where i ∈ [0..1] do
10             if distance(p, p_neighbor) < β then
11                 repulse(p, p_neighbor)
12                 removeIntermediateWaypoint(p)
13         // security check
14         for each pedestrian p_neighbor in f_p[i]
15         where i ∈ [1..n] do
16             select intermediate waypoint:
17             l_p[i − 1] or r_p[i − 1]
18             slowDown(p)
```

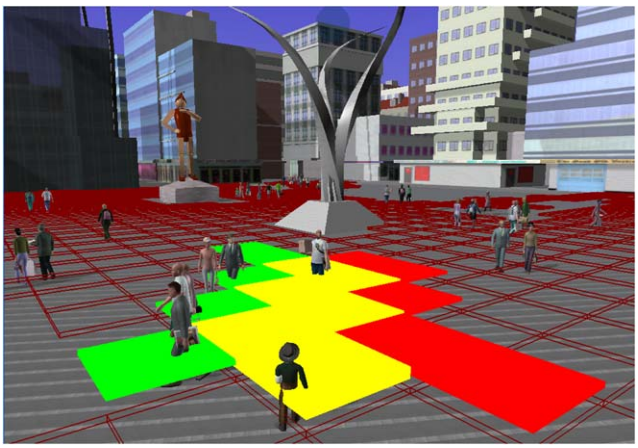


Fig. 4 A corridor of cells where the pedestrian looks for potential collisions. *Yellow cells* are in its front vector, *green ones* in its left vector, and *red ones* in its right vector. The number of cells depends on the speed of the pedestrian

Security check Once the cells to check have been identified, care is taken to prevent collisions. This is achieved as detailed in Algorithm 1: at line 14, we iterate over cells in front of p, starting from the second one, i.e., $f_p[1]$. We consider security checks in $f_p[0]$ unnecessary, because neighbors within this cell will require an emergency collision avoidance, which is treated below. For each of these front cells, if a neighbor pedestrian $p_{neighbor}$ exists, an intermediate waypoint $p.intwp$, preceding p's original waypoint is introduced to prevent the collision. The intermediate waypoint is either set to the right or to the left of p (line 16), depending on two conditions:

1. Is $p.intwp$ in the same general direction as the original waypoint $p.wp$? We do not want the pedestrian to perform a U-turn. This condition is tested against both possibilities $l_p[i-1]$ and $r_p[i-1]$:

$$(p.wp - p) \cdot (p.intwp - p) > 0.0$$

2. Is $p_{neighbor}$ rather on p's left side? This condition allows to find the fastest way to avoid $p_{neighbor}$:

$$\left[(p_{neighbor} - p) \times p.fwd\right] \cdot y < 0.0$$

Condition 1 has precedence: if the left alternative returns true while the right one returns false, then p will go left towards $l_p[i-1]$. However, if the results of Condition 1 are the same for both the left and right alternatives, then the result of Condition 2 determines the solution: p will go toward $r_p[i-1]$ if false, or toward $l_p[i-1]$ if true. Finally, on line 18, p is slowed down to correctly avoid its neighbor. When the collision avoidance is over, the speed will increase again in order to maximize the pedestrian's progress.

Emergency check An emergency avoidance is sometimes required when the previous approaches (potential fields, security check) fail. Such emergency cases also happen in reality, when people bump into each other because of a wrong evaluation of distance or speed. To prevent virtual pedestrians from passing through each other, we check in cells close around p that a minimum distance is always respected. This is shown in lines 5 to 12 in Algorithm 1. First of all, note that the check is performed over the two first rows of cells for all three vectors, i.e., $f_p[0,1]$, $l_p[0,1]$, $r_p[0,1]$ (line 8). Then, for each pedestrian registered in these cells, if the distance between p and $p_{neighbor}$ is smaller than a constant value β, both characters are moved away from each other, as if bumping against each other. Finally, in line 12, we avoid unnecessary detours by removing the intermediate waypoint if there was one, since it is no longer accurate: both pedestrians have been pushed away, and the situation has changed.

To keep the algorithm fast, we alternate the tasks of subscribing pedestrians in their cells and the actual avoidance: the pedestrians are registered in their cell every other time step (line 1), while the search for potential collisions and their avoidance is achieved at the next time step (line 4). Given the low distance covered by a pedestrian in such a short time lapse, the algorithm robustness is guaranteed.

4.5 Steering

Both navigation graph and potential field approaches provide waypoints which pedestrians must reach. A smooth steering algorithm is necessary to obtain a fluid movement toward these points. Reynolds' seek behavior [24] has the advantage of producing a believable steering toward a target point in space. We use this model for pedestrians of both ROI 0 and ROI 1, and a linear steering in ROI 2.

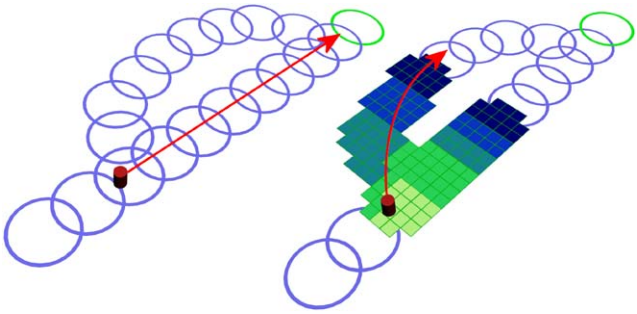


Fig. 5 (*Left*) In graph space, the path followed by the pedestrian is the right one. (*Right*) In grid space, the potential field is lower on the left path. High potential is represented in *light green* and low potential in *dark blue*

4.6 Continuity maintenance

In our architecture, we use two approaches based on different spaces: a navigation graph composed of vertices and edges, and a grid of cells. This duality brings up two issues when switching from one space to the other. More precisely, when a pedestrian passes from ROI 0 to ROI 1.

The first issue arises when a pedestrian enters the active grid space (ROI 0). Its position is then only updated in the grid, but no longer in the graph. It implies that this character stays registered in the same vertex while progressing in the grid. Thus, its next waypoint on the graph also remains the same. When the pedestrian eventually exits ROI 0, it turns back to meet the graph waypoint it has long since passed. To avoid this, we update the pedestrian position in the graph, even in ROI 0: if a pedestrian enters this region, we keep track of its distance to its next graph waypoint. When the distance is under a given threshold, the pedestrian is registered in the next vertex.

The second issue occurs when two or more paths of the same navigation flow are present in ROI 0. Since path planning in that area is ruled by the potential field, a pedestrian chooses the path where the potential is the lowest, as in Fig. 5 (right). However, this path does not necessarily correspond to the one it is registered to in the graph (Fig. 5 (left)). In the worst case, the pedestrian becomes completely lost when exiting ROI 0: it is within a vertex that does not belong to the path it should follow. To solve this problem, when a pedestrian exits ROI 0, we test whether it still is on the same graph path. If not, we look for a new path using this vertex and register the pedestrian to it.

5 Group behavior

Section 4 details how our scalable motion planning architecture handles individuals in different ROI. However, in our everyday life, it is rare to observe people in an urban scene walking all by themselves. Indeed, it is easy to notice that

pedestrians often evolve in groups of 2 or more. For this reason, we introduce an additional and optional layer to the architecture detailed in Sect. 4. Our algorithm is completely based on the update of pedestrian waypoints, similarly to the short-term avoidance approach. This choice is based on several criteria: firstly, all the motion planning algorithms used in this architecture are waypoint-based, which makes our grouping scalable to all ROI. Secondly, using a potential field to steer pedestrians close to each other is not possible; it would require a potential field computation for each small group, instead of each set, as introduced in Sect. 4.3. Moreover, it would not solve the scalability issue for ROI 1 and ROI 2.

At initialization, groups are created in the numbers and sizes provided by the user. Typically, they are composed of 2 to 5 pedestrians. For each of these groups, the first member is identified as the leader. The followers are assigned to the same path flow as the leader, they are registered in the same initial vertex, at the same speed. Note that special care is taken not to use the same human template twice in the same group in order to avoid a too important feeling of resemblance. Each created group is assigned a unique ID, and is stored as a list containing all its members' IDs.

At runtime, several consecutive operations are called. First, the waypoint of each pedestrian is updated according to its ROI if necessary. Then, the short-term avoidance is handled, potentially introducing an intermediate waypoint to prevent collisions. This algorithm remains the same for groups, except that members of a same group do not trigger a security avoidance. Otherwise, it would imply that group members try to avoid each other and remain together at the same time. The emergency check however is kept in order to inhibit inter-penetration. Once the short-term avoidance is achieved, groups are handled on an extra layer.

Speed adaptation For group members to stay close to each other, it is important that they react to the others' behavior. We start by computing the distance between them projected on the leader's forward vector:

$$leader.fwd \cdot distance(leader, follower). \tag{5}$$

This value provides the distance between the members in the direction of the leader's forward vector. If this distance is larger than a given threshold (in our implementation: 0.5 m), then the follower is too far behind and needs to catch up. The speed of the leader is thus decreased, while the follower's one is increased. On the other hand, if the result of (5) is smaller than the negative threshold (−0.5 m), the leader accelerates while the follower slows down. In the case where the distance is within the correct range, care is taken that both speeds are set to the same intermediate value. This is how we ensure that two members of a group remain at the same level. In order for their lateral distance to remain small, their waypoints are updated.

Waypoint modification It is important to note that in ROI 1, we only update the pedestrian's gate waypoints, i.e., waypoints situated on the intersection of two vertices. An intermediate waypoint that may be introduced during a security check is not modified. Thus, if two group members get dangerously near other pedestrians, they can separate to prevent a collision. They will get back close to each other by the time they reach their next gate waypoint. In ROI 0, the system is slightly different. At all times, the follower's waypoint is updated to be set close to the leader's (in a neighbor cell). However, if a security avoidance happens, then, similarly to ROI 1, waypoints are modified according to the algorithm of Sect. 4.4. Thus, group members can become separated. Once the potential collision is avoided, the follower's next waypoint is set back next to the leader's. In ROI 2, no short-term avoidance is performed, and members of the same group have the same speed and their waypoints are set close to each other on the vertice gates.

Note that for groups of 2 members, it is important that both members adapt their speed to stay together. In larger groups, however, we do not update the leader's speed according to the other members' positions. We then consider that the leader sets the pace and followers adapt their speeds to catch up.

6 Results and discussion

Our performance tests have been run with an Athlon64 4000+, with 2 GB RAM and two NVidia 6800 ultra in SLI mode. For these tests, pedestrians have no grouping behavior, and are represented with two human templates using several textures, and exploiting color variety techniques [3]. They are rendered as impostors, and use a walk animation, sampled at a frequency of 20 Hz. Note that in the following performance tests, we observe interesting emergent behaviors, e.g., lane formations or panic effects, that make the crowd motion planning more realistic. The tests described below are illustrated in the first accompanying video.

We use a city pedestrian area (Fig. 1 (right)) to test the performance of our motion planning architecture, compared with our implementation of the purely potential field-based approach [29]. In this scene, the camera position is fixed at a predefined position. For our tests, we define three regions in the environment. The one with the highest level of interest (ROI 0) has a radius of 15 m, and is static, in the center of the scene. The remaining space inside the view frustum is of low interest (ROI 1), and the other zones are classified in ROI 2. We have tested the efficiency of both approaches with cells of 3×3 m^2, and an increasing number of pedestrians and sets, starting from 2 sets and 200 pedestrians up to 12 sets, totaling 1200 characters. Figure 6 shows the results of this comparison. The performance of our approach logically decreases with the increasing number of sets, but much more

slowly than with the purely potential field-based approach. There are two reasons. Firstly, our technique only computes the potential field in a limited region of high interest (ROI 0). Secondly, only a subset of the total number of sets passes in this region, minimizing the number of potential fields to compute. This test has also been performed with the ROI 0 dynamically moving on the city place (demonstrated in the first accompanying video). Even so, the obtained results remain similar to those shown in Fig. 6.

Our second test is achieved with 10,000 pedestrians in a large scene with 12 navigation flows, i.e., 24 sets, spread over the whole environment, as demonstrated in Fig. 1 (left). For this scenario, the ROI are placed according to the camera position. If the camera moves, the regions are also displaced. The cell size is set to 4×4 m^2, and the performance reaches 20 fps.

The third scenario uses the same city pedestrian area as in the first test, extended with several surrounding streets and buildings. There are 5000 pedestrians and some cars on the roads, as illustrated in Fig. 7 (left). Each cell of the grid covers a 3×3 m^2 area. Since the user attention is mainly drawn by the threatening cars, a region of high interest (ROI 0) is set around each of them. Moreover, to make pedestrians flee potential collisions, a high discomfort and speed increase are set in front of the cars, as in [29]. As a result, pedestrians close to a car are always in a region of high interest, and thus ruled by a potential field. In front of cars particularly, the pedestrians flee the zone of danger, demonstrating an emergent panic behavior. The remaining visible environment is classified as a region of low interest (ROI 1), so that pedestrians still take care to avoid each other, while the zone outside the view frustum is set as of no interest (ROI 2). The resulting fps varies between 15 and 30, depending on the number of visible cars (1 to 3), and the size of their surrounding ROI 0 (10 to 15 m radius). We note that the use of a single walk animation in our companion video generates foot sliding artifacts when pedestrians greatly increase their speed. However, we do not focus on the character animation accuracy, but rather on the global crowd behavior.

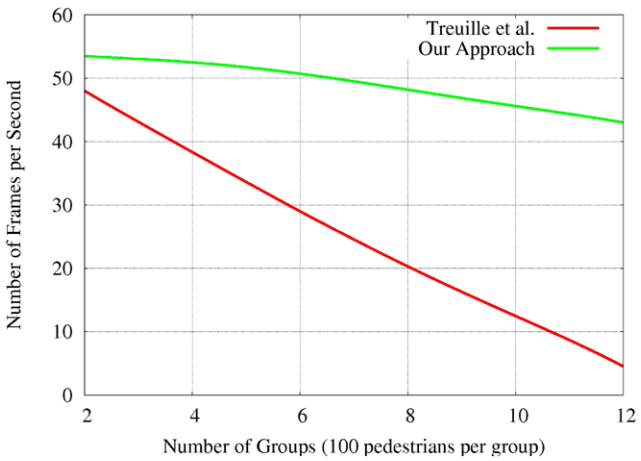


Fig. 6 Comparison between our approach and our implementation of the approach of Treuille et al. [29] for a varying number of sets, each composed of 100 pedestrians

Finally, we have tested the frame rate evolution with a fixed number of 24 sets and an increasing number of pedestrians. The test has been conducted in a large scene with cells of 3×3 m^2, and 1 to 5 distinct ROI 0, each of a 15 m radius. For the remaining of the scene, ROI 2 is not exploited; all vertices are classified as ROI 1. During the test, the scene and pedestrian rendering are deactivated to analyze the sole motion planning cost. The results, in Fig. 8, show that even with 5 distinct ROI 0, our architecture manages the motion planning of 10,000 pedestrians at interactive frame-rates (between 10 and 15 fps). Note that the increasing number of pedestrians does not influence the potential computation, which is more sensible to the number of sets, as much as the short-term avoidance, which has a complexity in $O(n^2)$.

In order to assess our implementation of small group behaviors, we have created eight short movies that have been shown to twelve subjects with no Computer Graphics background. Four of the movies represent a pedestrian city center (Fig. 1 (right)), while the other four were shot in a Halloween theme park (Fig. 7 (right)). For these environments, each of the four movies has a specific scenario: *alone*, where pedestrians only walk by themselves; *small*, where pedestrians are in groups of 2 only; *large*, for large groups of 3

Fig. 7 (*Left*) A city scene where pedestrians avoid a car surrounded by a ROI 0. (*Right*) A Halloween theme park visited by families

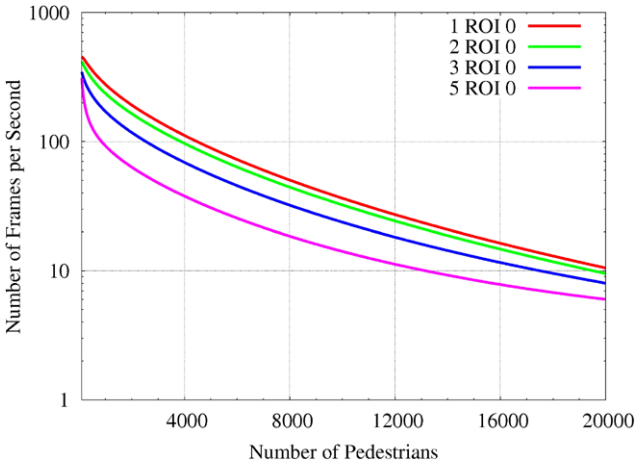


Fig. 8 Performance for 24 sets with an increasing number of pedestrians (no rendering). 1 to 5 ROI 0 of a 15 m radius each are placed in the scene, while the remaining space is entirely in ROI 1

to 4 people; and *mix* for a varied distribution of size from 1 to 4 group members. Note that the second video accompanying this paper is an illustrative montage based on the ones used for testing.

We asked the twelve subjects to watch the videos and pay particular attention to the pedestrians and their behavior. They were all shown the eight videos of 30 seconds each in different order to avoid a bias in the answers. After each viewing, several questions were asked:

- Indicate what you like/dislike about this scene.
- Who are these people and what are they doing?
- Do they look sympathetic?
- Do you feel at ease?

We intentionally hid the topic of the study and asked questions not directly concerning the groups to get a feeling of subjects' reactions, and to see if they would notice the different scenarios. In the movies, the human templates used were 3 adult women, 2 adult men and 1 boy. Rendering variety techniques were employed to further diversify the crowd [3].

Contrary to our expectations, only a small number of subjects directly noticed that the size of groups was changing with the videos. However, through naive comments, they showed that the feeling was different for changing scenarios. For the city center, in the *alone* scenario, people found that pedestrians looked stressed out, in a hurry, going to work. Especially concerning the child template, some people wondered where his parents were, or assumed he was going home from school. Several people thought that the child was in fact a person of short stature when no adult was accompanying him. However, when groups were introduced, the subjects talked about an easy-going ambiance, less strict. The few lonely pedestrians in the *mix* scenario were going to work, the groups were visitors, tourists, families, or friends shopping together. In the Halloween theme park, the same type of comments were observed. In the *alone* scenario, people thought that the point of view was close to the exit. People noticing lonely children expressed the same feeling as in the city. In the grouped scenarios, pedestrians were seen as shoppers, families and friends, or visitors. In the *mix* scenario, pedestrians walking alone were interpreted as park staff. These results show that it is possible to completely modify a scene atmosphere by introducing groups of different sizes.

7 Conclusion and future work

This paper presents a hybrid architecture allowing realistic real-time crowd motion planning for thousands of pedestrians. Our approach is scalable; it is possible to divide the scene into regions and exploit different motion planning algorithms according to their level of interest. The architecture flexibility allows the user to determine the performance he wishes and to select and distribute the regions of interest (ROI) accordingly. Also, we show that it is possible to insert an additional and scalable layer to simulate small groups of pedestrians.

Our implementation employs an accurate potential field-based method for pedestrians in ROI 0. A simple and efficient short-term avoidance algorithm is exploited in both ROI 0 and ROI 1, thus ensuring no noticeable transition at region borders. Results show that it is possible to simulate over 10,000 characters in real time, while defining many more sets than in a purely potential field-based approach. Realism is further demonstrated with observable emergent behaviors, like lane formations and panic escape. Finally, a simple study has been conducted in order to assess our group implementation, and shows that such an addition offers a larger variety of interpretations, and a more sympathetic ambiance in the scenes.

There are some limitations to our architecture. Firstly, in too crowded narrow environments, severe bottlenecks may appear, making the use of our potential field-based approach a waste of computational time. However, it is possible to enforce a low level of interest in these regions, e.g., ruled by the short-term avoidance algorithm introduced in this paper. Another limitation is our set-based approach: we are constrained to assign general goals for sets of pedestrians. One goal per pedestrian would be too prohibitive for real-time applications. Yet, we note that our architecture can handle many more sets than previous potential field-based methods. This is mainly due to our massive reduction of the number of cells in which the potential is actually computed, and implies the possibility to refine the grid for more accurate results.

An interesting lead for future work is to investigate agent-based algorithms to improve pedestrian behaviors, and possibly merge them with our ROI architecture. Moreover,

based on our study, we have gathered many interesting leads to improve group behaviors: care should be taken when choosing group members to improve realism, e.g., a businessman wandering with a casually clothed person may raise questions, children should be accompanied with at least one adult, etc. Also, interactions between group members and among different groups should be introduced. For instance, children should occasionally look at their parent, group members should talk to each other, people from different groups could show interest in their surroundings, maybe recognize other people from time to time and wave.

Acknowledgements We would like to thank Mireille Clavien for her exceptional work on designing the virtual humans and the scenes. Many thanks to Helena Grillon for proofreading this paper. This project has been sponsored by the Swiss National Research Foundation.

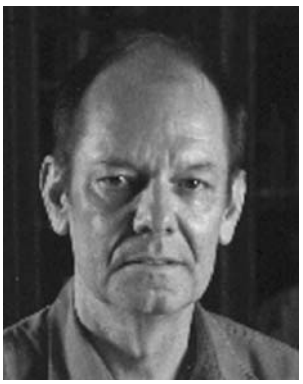
References

1. Bayazit, O.B., Lien, J.M., Amato, N.M.: Better group behaviors in complex environments using global roadmaps. In: ICAL 2003, pp. 362–370 (2003)
2. Chenney, S.: Flow tiles. In: SCA'04, pp. 233–242 (2004)
3. de Heras Ciechomski, P., Schertenleib, S., Maïm, J., Maupu, D., Thalmann, D.: Real-time shader rendering for crowds in virtual heritage. In: VAST'05, pp. 1–8 (2005)
4. Heïgeas, L., Luciani, A., Thollot, J., Castagné, N.: A physically-based particle model of emergent crowd behaviors. In: Graphicon (2003)
5. Helbing, D., Molnár, P., Schweitzer, F.: Computer simulations of pedestrian dynamics and trail formation. In: Evolution of Natural Structures, pp. 229–234 (1994)
6. Helbing, D., Farkas, I., Vicsek, T.: Simulating dynamical features of escape panic. Nature **407**(6803), 487–490 (2000)
7. Hughes, R.L.: A continuum theory for the flow of pedestrians. Transp. Res. Part B Methodol. **36**(29), 507–535 (2002)
8. Hughes, R.L.: The flow of human crowds. Annu. Rev. Fluid Mech. **35**(1), 169–182 (2003)
9. Kamphuis, A., Overmars, M.H.: Finding paths for coherent groups using clearance. In: SCA'04, pp. 19–28 (2004)
10. Kirchner, A., Shadschneider, A.: Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. Physica A **312**(1–2), 260–276 (2002)
11. Lamarche, F., Donikian, S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. Comput. Graph. Forum **23**(3), 509–518 (2004)
12. Lau, M., Kuffner, J.J.: Precomputed search trees: Planning for interactive goal-driven animation. In: SCA'06, pp. 299–308 (2006)
13. Lee, K.H., Choi, M.G., Hong, Q., Lee, J.: Group behavior from video: A data-driven approach to crowd simulation. In: SCA'07 (2007)
14. Loscos, C., Marchal, D., Meyer, A.: Intuitive crowd behaviour in dense urban environments using local laws. In: TPCG'03, p. 122 (2003)
15. Metoyer, R.A., Hodgins, J.K.: Reactive pedestrian path following from examples. In: CASA'03, p. 149 (2003)
16. Morini, F., Yersin, B., Maïm, J., Thalmann, D.: Real-time scalable motion planning for crowds. In: Cyberworlds International Conference, pp. 144–151 (2007)
17. Musse, S.R., Thalmann, D.: A model of human crowd behavior: Group inter-relationship and collision detection analysis. In: Eurographics Workshop on Computer Animation and Simulation (1997)
18. Niederberger, C., Gross, M.H.: Hierarchical and heterogeneous reactive agents for real-time applications. Comput. Graph. Forum **22**(3), 323–331 (2003)
19. Paris, S., Pettré, J., Donikian, S.: Pedestrian steering for crowd simulation: A predictive approach. In: Eurographics'07 (2007)
20. Pelechano, N., Allbeck, J., Badler, N.: Controlling individual agents in high-density crowd simulation. In: SCA'07 (2007)
21. Pettré, J., de Heras Ciechomski, P., Maïm, J., Yersin, B., Laumond, J.P., Thalmann, D.: Real-time navigating crowds: scalable simulation and rendering. J. Vis. Comput. Animat. **17**(3–4), 445–455 (2006)
22. Pettré, J., Grillon, H., Thalmann, D.: Crowds of moving objects: Navigation planning and simulation. In: ICRA'07 (2007)
23. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: SIGGRAPH'87, pp. 25–34 (1987). DOI: http://doi.acm.org/10.1145/37401.37406
24. Reynolds, C.: Steering behaviors for autonomous characters (1999)
25. Reynolds, C.: Big fast crowds on ps3. In: Sandbox'06: Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames, pp. 113–121 (2006)
26. Shao, W., Terzopoulos, D.: Autonomous pedestrians. In: SCA'05, New York, NY, USA, pp. 19–28 (2005)
27. Sung, M., Gleicher, M., Chenney, S.: Scalable behaviors for crowd simulation. Comput. Graph. Forum **23**(3), 519–528 (2004)
28. Sung, M., Kovar, L., Gleicher, M.: Fast and accurate goal-directed motion synthesis for crowds. In: SCA'05, pp. 291–300 (2005)
29. Treuille, A., Cooper, S., Popovic, Z.: Continuum crowds. In: SIGGRAPH'06, pp. 1160–1168 (2006). DOI: http://doi.acm.org/10.1145/1179352.1142008

Barbara Yersin is a research assistant and PhD candidate at VR-LAB, EPFL. She receives her Master Degree in Computer Science at EPFL in 2005, after performing her Master Project at the Université de Montréal. Her main interests are in real-time applications, particularly crowd simulations. The subject of her PhD Thesis is the real-time motion planning and behavior of crowds of virtual humans.

Jonathan Maïm is a research assistant and PhD candidate at VR-lab at the Swiss Federal Institute of Technology in Lausanne (EPFL). In April 2005, he receives a Master Degree in Computer Science from EPFL after achieving his Master Project at the University of Montréal. His research efforts are concentrated on building an architecture for simulating real-time crowds of thousands of realistic virtual humans.

Fiorenzo Morini is a research assistant at the University of Applied Sciences of Southern Switzerland (SUPSI). In February 2007, he receives a Master Degree in Computer Science from EPFL, after having completed his Master Project in the field of avoidance in virtual crowds. His research interests focus on real-time avoidance algorithms applied to large groups of virtual humans.



Daniel Thalmann is Professor and Director of The Virtual Reality Lab (VRlab) at EPFL, Switzerland. He is a pioneer in research on Virtual Humans. Daniel Thalmann has been Professor at The University of Montreal and Visiting Professor/ Researcher at CERN, University of Nebraska, University of Tokyo, and National University of Singapore. He is the President of the Swiss Association of Research in Information Technology and one Director of the European Research Consortium in Informatics and Mathematics (ERCIM). He is coeditor-in-chief of the Journal of Computer Animation and Virtual Worlds, and member of the editorial board of 6 other journals. Daniel Thalmann was member of numerous Program Committees, Program Chair and CoChair of several conferences. He has also organized 5 courses at SIGGRAPH on human animation and crowd simulation. Daniel Thalmann has published numerous papers in Graphics, Animation, and Virtual Reality. He is co-editor of 30 books, and co-author of several books including "Crowd Simulation", published in 2007 by Springer. He received his PhD in Computer Science in 1977 from the University of Geneva and an Honorary Doctorate (Honoris Causa) from University Paul-Sabatier in Toulouse, France, in 2003.