Vision Aware Continuum Crowds

Wilfrid Lefer
University of Pau
64 013 Pau, France
lefer@univ-pau.fr

ABSTRACT

Crowd simulation has received increasing attention for two decades because potential applications of crowd simulators can be found in various societal domains. The continuum crowd model puts all information related to the decision-making process in a single equation, which can then be solved by a Fast Marching Method approach. In this paper we propose several improvements to the continuum crowd model: a new governing equation, a new collision avoidance method, and, our major contribution, we add vision capabilities to the characters, thus making them able to collect new information about their surrounding environment and to reconsider path planning according to up-to-date data.

Keywords

Computer animation, crowd simulation, continuous models.

1. INTRODUCTION

Crowd simulation has received increasing attention for two decades because potential applications of crowd simulators can be found in various societal domains: architecture, robotics, building evacuation, training systems, video games, etc. The problem has been addressed from various viewpoints, depending on the original background of people tackling it: architecture, physics, psychology, computer graphics, etc. Thus various issues require attention, although rarely all of them are addressed at the same time, ie in a single system: virtual human generation, character/rigid body animation, crowd simulation, virtual crowd rendering, and even interacting with virtual crowds. Rather than listing more or less exhaustive reference list, the reader may start with a recent dedicated book [Tha07]. A number of crowd simulators are now available, both commercial solutions and academic ones. Basically they can be divided into two categories: behavior versus rendering oriented. The first category of systems try to generate realistic behavior of crowds, generally for specific environments and simulations, such as building evacuation, the objective being to assess emergency procedures in realistic emergency conditions, that is with people in panic and so. The second category is mainly devoted to the games industry and systems must be able to simulate large crowds wit drastic frame rate constraints. Here rendering quality is more important than realistic behavior.

Several crowd models can be found in the literature, the most popular ones being rule-based systems, Reynolds's social forces [Rey87], and recently continuum crowd [Tre06]. More sophisticated models propose a layered architecture so as to consider different information abstractions and hence different behavior modeling levels, such as reactive versus cognitive [Bra03][Pel07]. Basically crowd simulation models can be divided as either discrete or continuous. In a discrete model, a number of parameters are evaluated at each time step and a decision is taken, for instance move forward or turn on the right or stay here. A small change of a single parameter can lead to a radically different decision. Typical such models are rule-based ones in which each possible action is predicated by a condition, all possible actions being tested in a given order until a valid one had been found. In a continuous model, a moving direction is computed as a – continuous - function of all the parameters. Thus a slight change of a parameter leads to a slight change of the moving direction. The continuum crowd method is probably the more elaborated such model [Tre06]. This paper extends this model by adding vision capabilities to the crowd.

The remaining of this paper is organized as follow. Section 2 recalls the main elements of the continuum crowd model, together with some improvements. Section 3 presents our solution to add vision skills to crowd characters. Some results and a brief discussion on the limits of our solution are presented in section 4 and section 5 is our conclusion.

2. CONTINUUM CROWD REVISITED

The basic issue any crowd simulation system has to address at each time step of a simulation is to

compute the next displacement of each character, that is the moving vector (direction and distance). Various data can be integrated in the decision-making process at this time but specific data are considered in any crowd simulator: physical environment, that is geometry (height, obstacles), and positions of other characters. The continuum crowd model considers 3 kinds of information: terrain slope, crowd density, and discomfort. All of them are continuous functions of $\Re^2$, i.e. the 2.5D simulation space. The discomfort field is a flexible way to design how comfortable is an individual at a given location. Thus it is possible to make characters that will more likely walk in pedestrian areas rather than on car drives. Obstacles are not explicitly defined but they can be easily described their impact on the discomfort field, typically by assigning high discomfort values to obstacles locations.

New equation

The principle of any path search method consists in minimizing a travelling cost C, which computation can involve various parameters:

$$\text{Min}[\int_P \mathbf{C} ds)] \qquad (1)$$

where P is the path and ds means that integration is taken with respect to path length.

This requires to be able to evaluate the cost function at any point for all possible directions, which yields to a function of two variables: location and direction of movement. The original continuum crowd cost function is as follow [Tre06]:

$$C(x,\theta) = (\alpha f + \beta + \gamma g) / f \qquad (2)$$

Where x and θ are the location and direction of movement, respectively, f is the speed at location x, g is the discomfort at position x+θ, and α, β, and γ are weights (values in the range [0,1] and verifying condition $\alpha + \beta + \gamma = 1$) for individual terms, which can be used to define the relative influence of each term on the final cost.

This equation raises some problems though, because it is not linear in the weights, which makes it difficult when an accurate control of the displacement times is required, for instance for the purpose of building evacuation certification. Another issue arises for very small values of f, for instance in congestion situations.

For this reason we will use a slightly different function:

$$C(x,\theta) = \alpha + \beta\,(1 - [(f(x,\theta) - f_{min}) / (f_{max} - f_{min})]) + \gamma\, g(x+\theta) \qquad (3)$$

This yields to expected results at limit conditions. For instance if we neglect speed (β=0) and discomfort (γ=0), the unit cost is 1, which yields to the shortest path. If distance (α=0) and discomfort are neglected, we minimize travelling time. F_{min} and F_{max} are the minimal and maximal speeds, which correspond to extreme situations, typically in congestion and free run at maximum negative slope, respectively. An interesting property of this function is that it generates values in the range [0,1].

Because equation 1 cannot be solved analytically, a discrete version is used, which allows us to compute an approximation of the optimal path. Thus the simulation space is discretized as a regular grid and each field (height, speed, discomfort, cost) is evaluated at each cell center for isotropic fields (height, discomfort) or using a MAC-style arrangement [Fed01] for anisotropic ones.

Collision avoidance versus minimum distance enforcement

The continuous continuum crowd model can prevent collisions, assuming density threshold is properly set. However, approximation solutions of equations 1 in discrete space lead to numerical errors, a typical consequence being that collisions actually occur. Treuille et al. propose a minimum distance enforcement mechanism to address this issue. It consists in explicitly checking all pairs of individuals in the vicinity of a character before moving it. The overhead involved by these tests can be important, especially in congestion situations.

We adopt a different strategy. Once theoretical displacements have been computed by the simulator, an advector is in charge of actually moving the individuals. Several specialization classes have been design, which all inherit from a generic advector class and implements different advecting strategies, the most basic one consisting in just advecting the crowd. But more sophisticated advectors have been developed, which allows us to evaluate different collision avoidance strategies.

One of them implements the following algorithm:

```
AdvectCrowd(list<character> L)
{
    While L is not empty Do
        Extract a character from the list randomly
        Vector v = theoretical displacement vector
        Position p = character position + v
        Integer #tries = 3
        While #tries > 0 And there is a collision Do
            v = v / 2
            p = p – v
            #tries = #tries – 1
        If #tries > 0 Then
            Move character to its new position
}
```

Changing the order of advection from a time step to the next is important because it contributes to keep crowd compactness and it avoids strong slow downs. This can arise for instance in a tight corridor in which overtaking is impossible. In such a situation, if it turns out that the individual at the head of the queue is advected in last position, all other characters will be blocked, their advections being impossible because there is no space in front of them. Note that except in such special situations, this kind of problem is properly addressed by the simulation phase, the high density in front of an individual will make it try to overtake by the right or by the left.

This algorithm offers a good trade-off between advection quality and computation complexity. Quality of advection can be measured as the distance between positions of the crowd using the theoretical advection, that is displacement vectors computed by the simulator, and actual positions, i.e. collision free positions.

3. VISION AWARENESS

Here we detail how individual characters of our continuum crowd model get vision capabilities and thus are able to react to changes in their visual environment.

Problem statement

In the original continuum crowd model [Tre06], every individual has a complete knowledge of the whole environment and is able to determine an optimal path toward his own group target. Obviously this model is not realistic in many common real world situations. A trivial case is a discovery process, in which someone is trying to find his/her way in a partially unknown environment. Another common situation is a dynamic environment, for instance a city in which a street becomes suddenly no longer available for some reason: people far from this place are typically not aware of this event and hence should not reconsider their path.

A realistic model should take individual knowledge into account, rather than collectivity knowledge. Moreover this knowledge should be subject to changes, as a function of the various events encountered by each individual, typically events occurring in his/her vision space. This implies for a simulator to include the following mechanisms:

- manage individual knowledge, that is each individ*ual has his own knowledge base,*
- handle knowledge evolution over time.

Individual knowledge representation

The original continuum crowd model does not consider the large variety of information that can be available in an environment, such as buildings, roads, trees, lakes, fire, hard-to-walk-in places such as dense forests or marsh, etc, but rather considers discomfort as a way to take any kind of information into account. A nice property is that equation 2 only involves three data fields: height, people density, and discomfort. Up to the end user to design any function to map the various information data to discomfort.

In order to have each individual reasoning with his/her own knowledge base, it is necessary to duplicate these three fields for every character. This is obviously memory consuming, especially for large crowds, which are indeed the main target of the original continuum crowd model. We decrease this memory overhead by storing only the cost field $C(x,\theta)$ instead (see equation 3).

Dynamic individual knowledge reconstruction

Now we describe how each individual knowledge base is updated at each time step of the simulation. At the beginning, people in a single group share a common knowledge base. Then at each time step, each individual updates his/her knowledge base according to his/her percepts of the environment. Percepts can be simply visual or they may include information gathered by any source, such as voice diffusion – a fire alert for instance – or mobile phone communications. For each kind of percept, a specific information acquisition mechanism, called percept processing unit, is set up, which affects some cells of the simulation grid. Terrain slope being likely not to change over time, only the crowd density and discomfort fields are affected by the percepts processing unit. Then $C(x,\theta)$ is updated only for the cells affected by the percepts. Last, the Fast Marching Method is run in order to produce a new potential field, which will give the moving direction for the avatar.

Vision percept computation

Even if various perception channels can be used to improve individual knowledge, we concentrate here on the visual acquisition of information. This implies to be able to determine which part of the scene is visible by a given character at any time. The data that can be perceived are stored in the simulation grid: people positions, moving obstacles such as cars, smoke, etc. Hence determining which information is directly visible by a character consists in determining the list of grid cells that are directly visible from the character's position. A trivial way to solve this problem consists in processing every obstacle in order to determine the shortest visible distance in each possible direction. For geometric obstacles, this can be typically implemented with a Z-buffer algorithm. But such an approach is not appropriate in the case of the our continuum crowd scene for the following reasons:

- there is no longer geometric obstacles but rather they have contributed to determine discomfort values (typically walls correspond to infinite discomfort values),
- we need the visibility region as a list of cells and thus we would have to compute this list from the Z-buffer values, which would had a computation overhead,
- the process could be time consuming for a large number of obstacles, while only a few of them is generally visible from a given location.

Instead we formulate the visibility condition as the solution of two separate Eikonal equations [Set99a]:

$$|\nabla T_{\text{no-obstacles}}| = 1 \qquad |\nabla T_{\text{obstacles}}| = f(x,y)$$

with $f(x,y)=\infty$ where $g(x,y)=\infty$ and $f(x,y)=1$ anywhere else.

Remember that the discomfort field $g(x,y)$ is everywhere positive and equal to ∞ for cells that fall inside an obstacle, typically a wall. Solving this equations system consists in solving each equation separately and then comparing results cell by cell. For a given cell, two strictly equal results means that the cell center is visible from the character position, otherwise the cell center is not visible. To understand this, remember that solving the Eikonal equation amounts to computing the optimal path from the source to the target. Visual rays use the straight line, which means that solution of the left equation corresponds to the straight line. Solution to the right equation computes the optimal path by treating $f(x,y)$ as a cost function, the time necessary to travel through a given cell being inversely proportional to the cost associated to that cell. Hence the only case for which both equations lead to the same value if when all cells on the straight line have the minimum cost of 1.

The implementation of this algorithm is not trivial, so we skip it here, the reader will find all details on the Fast Marching Methods implementation in [Set99b]. Now we suppose that there exists a function called *ComputeVisibilityList(c0 :<cell>)*, which computes the visibility cell list for a given character position c0.

What we need actually to update each character's knowledge base is the list of cells whose information has changed since the last time they have been processed by this character. So we withdraw from the visibility list the cells that were already in the visibility list at the previous time step and whose attached values have not changed at the current time step.

The algorithm for updating individual knowledge bases is as follows:

```
<list of cells> DetermineListOfCellsToUpdate(c0 : <cell>)
{
    Vt : visibility list at time step t
    It(c) : information stored in cell c at time step t
    c : <cell>
    Vtemp, Vchange : <lists of cells>
    Vtemp = Vt
    While Vtemp is not empty Do
        c << Vtemp
        If It(c)≠ It+θ(c) Then
            Vchange << c
    Vt+θ = ComputeVisibilityList(c0)
    Vθ = Vt+θ - Vt + Vchange
    Return Vθ
}
```

Once cells which cost changed at the last time step have been identified, we update the potential field by running the Fast Marching Method on the updated cost field C.

4. RESULTS

Figure 1 shows a simulation of two avatars moving in a building. Since we are just concerned by their trajectories, a basic 2D rendering is used here. The blue avatar implements the original continuum crowd algorithm whereas the magenta one is managed by our vision aware method. Both characters start from the same location at the same time (frame #0). Suddenly (frame #34), a fire arises at a place hidden from the characters' positions. The fire is modeled by increasing discomfort at the corresponding location. More precisely a discomfort footprint with a Gaussian shape is added to the discomfort field so as to not only prevent people from crossing the fire but also to simulate the heat effect, which should ideally make people passing at a reasonable distance from the fire spot. We can observe (frame #38) that, as soon as the fire starts, the blue character changes his trajectory and finds a more appropriate path, though not viewing the fire, as if he would be immediately aware of this event. The magenta character, who has visual capabilities, continues on the same path (frame #88). As soon as he is able to watch the fire (frame #100), he changes his path accordingly, which in this case yields to going back in the corridor (frame #108) and follow an alternative path (frames #180 and #280).

Our simulator can also provide a realistic 3D rendering with 3D characters, which are animated with the Cal3D library [Cal3D].

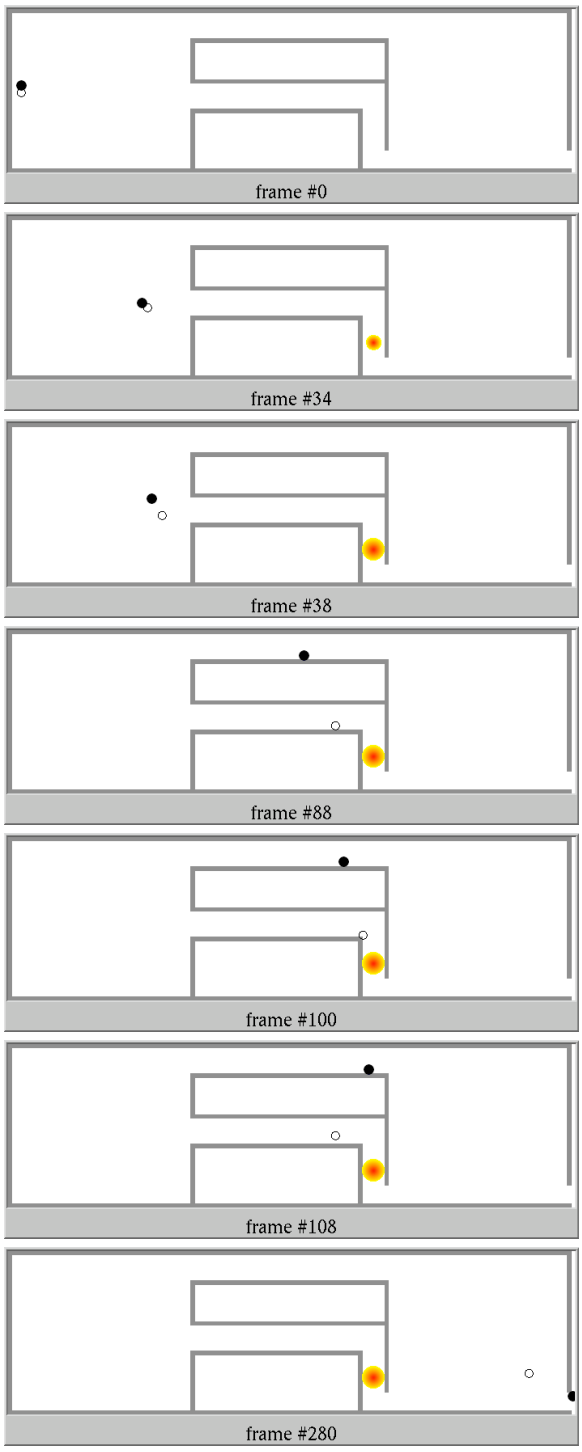

Figure 1: Original continuum crowd model (blue) versus vision aware model (magenta).

Limits

One of the nice properties of the original continuum crowd formulation is its scalability with the number of avatars. The memory and computational complexity is linear with the number of avatar groups, rather than the number of avatars. In large crowds of people having common objectives and common discomfort fields, computation is particularly efficient. Introducing variety in the discomfort as a function of information gathered along each individual path leads to having as many copies of the discomfort field as the number of avatars. This leads to both memory and computational overheads. But people starting at about the same location toward a common goal leads to a flocking phenomenon [Rey87]. As a consequence, they tend to observe the same phenomena at the same time. Such set of people might be grouped together as sub-groups, so as not to duplicate their discomfort fields. An error metrics could be defined in order to evaluate the correlation degree of individual knowledge within each sub-group, together with conditions for sub-group cutting.

Simulating heat effects of the fire is one of the properties that a Gaussian footprint can provide but of course this is not a reliable approach to model heat transfer: after a certain amount of time, heat should be felt at a distance that is not directly related to visibility. But another nice property is to avoid a geometric description of the fire, which would probably lead to people passing exactly at the limit of the fire zone, thus describing unrealistic perfectly circular trajectories.

5. CONCLUSION

Continuum crowds is an elegant model for crowd simulation, providing a mathematical and hence robust environment for various crowd behavior modeling. Our current work consists in evaluating the potential of this model for various crowd behaviors, which are typically modeled through a discrete formulation. Among them, vision awareness is an important issue since all entities that may be candidates for simulation do have vision capabilities: pedestrians, cars with their drivers, animals, even robots. Our solution is successful in enabling individuals to perceive their visual environment but adds an important computation overhead. Further investigations are required to find ways to decrease this cost, grouping individuals with similar viewpoints being a possible solution.

A further step in the direction of realistic crowd modeling needs to consider individual diversity in the way this information is processed in each brain and to model the underneath cognitive mechanisms. Coupling a cognitive reasoning module with a continuum crowd model in a hybrid architecture is one of our challenges now. One of the main issues is the continuous versus discrete models, cognitive models, such as those found in the Multi-Agent Systems literature, being generally discrete.

6. REFERENCES

[Bra03] Braun, A., Raupp Musse, S., and L.O.B. Bodmann. Modeling Individual Behaviors in Crowd Simulation. Computer Animation and Social Agents, New Brunswick (USA), 2003, pages 143-148.

[Fed01] Fedkiv, R., Stam, J., and H. Jensen, H. Visual Simulation of Smoke. In ACM Computer Graphics (SIGGRAPH'01 proceedings), pages 15–22.

[Pel07] Pelechano, N., , Allbeck, J., and N. Badler. Controlling Individual Agents in High-Density Crowd Simulation. ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'07), August 3-4, San Diego (USA), 2007.

[Rey87] Reynolds, C. W. Flocks, Herds, and Schools: A Distributed Behavioral Model. Computer Graphics 21(4) (SIGGRAPH'87 Conference Proceedings), pages 25-34.

[Set99a] Sethian, J.A., Level Set Methods and Fast Marching Methods, Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Material Science, 2nd edition, 1999, Cambridge University Press.

[Set99b] Sethian, J.A., Fast Marching Methods, SIAM Review 41, July 1999.

[Tha07] Thalmann, D., and S. Raupp Musse. Crowd Simulation. Springer, 2007.

[Tre06] Treuille, A., Cooper, S, and Z. Popovic. Continuum Crowds. ACM Transactions on Graphics 25(3) (SIGGRAPH'06 Conference Proceedings).

[Cal3D] http://home.gna.org/cal3d/