CA-LOD: Collision Avoidance Level of Detail for Scalable, Controllable Crowds

Sébastien Paris, Anton Gerdelan, and Carol O'Sullivan

GV2, Trinity College Dublin, Ireland
{Sebastien.Paris,gerdelaa,Carol.OSullivan}@cs.tcd.ie
http://gv2.cs.tcd.ie/metropolis

Abstract. The new wave of computer-driven entertainment technology throws audiences and game players into massive virtual worlds where entire cities are rendered in real time. Computer animated characters run through inner-city streets teeming with pedestrians, all fully rendered with 3D graphics, animations, particle effects and linked to 3D sound effects to produce more realistic and immersive computer-hosted entertainment experiences than ever before. Computing all of this detail at once is enormously computationally expensive, and game designers as a rule, have sacrificed the behavioural realism in favour of better graphics. In this paper we propose a new Collision Avoidance Level of Detail (*CA-LOD*) algorithm that allows games to support huge crowds in real time with the appearance of more intelligent behaviour. We propose two collision avoidance models used for two different *CA-LOD*s: a fuzzy steering focusing on the performances, and a geometric steering to obtain the best realism. Mixing these approaches allows to obtain thousands of autonomous characters in real time, resulting in a scalable but still controllable crowd.

Introduction

Our work contributes to the *Metropolis* project: an interdisciplinary effort involving computer graphics, engineering and neuroscience researchers aiming to develop a multisensory Virtual Dublin populated by scalable crowds with a high level of realism. Building on earlier work [1], the project is aiming to simulate large, scalable crowds while optimising the level of variety in animation, appearance and sound. Most relevant to our work, are the objectives to:

- Provide easily integrated populace and traffic for Virtual Environments that will be scalable: at the architectural level (i.e., from PC to large cluster exploiting multi-core

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 13–28, 2009.

architectures); the user level (i.e., from single user to massively multi-player); and the crowd level (i.e., thousands of people simulated in real-time).
- Increase the realism of these large crowds of people by adding variety in animation, appearance and sound, driven by perceptual models and metrics.
- Add real meaning to the simulations by endowing individual crowd members with appropriate, sentient behaviours that are based on cognitive and sociological models.
- Optimise the computational and memory resources necessary to achieve these real-time multisensory displays and ameliorate any consequential anomalies by applying principles of human perception and exploiting human perceptual limitations.

We have been investigating several ways in which to simulate plausible behaviours that take context into account [2]. In this paper we describe a candidate methods for simulating the navigation behaviours in Metropolis. Our aim is to push away the crowd behaviours' computational limitations without sacrificing the animation's realism and interactivity.

We first propose in *Section 2* a modular autonomous agent architecture focusing on the human's ability to move, and introduce the notion of Collision Avoidance Level Of Detail (*CA-LOD*). In *Section 3* we detail the fuzzy steering model that is used as a fast but unrealistic *CA-LOD*. In *Section 4* we propose another *CA-LOD* which improves the realism at the expense of the computation performance. Then in *Section 5* we evaluate the performance of each *CA-LOD*, and propose an extrapolation of the possible mixes to simulate huge crowds in real-time. We finally conclude in *Section 6* and present the promising future work for the presented framework.

1 Related Work

Animating large crowds of people in real time requires to address many problems: designing the simulation (either the virtual environment or the population), rendering the environment and the crowd, simulating the crowd behaviours, and animating the population to reflect its actions. Solutions to the rendering bottleneck are now well known, either by reorganising the scene-graph for efficient culling to improve static environments rendering [3], or by introducing graphical *LOD*s to improve the rendering of specific objects [4]. Specific solutions to crowds rendering also exist to greatly reduce the number of polygons to render each agent by applying pre-rendered textures to simple shapes using impostors or similar techniques [5]. All of these solutions focus on the computation done by the GPU while letting apart the behavioural algorithms which are generally executed by the CPU.

Indeed, animating hundreds to thousands people in real time requires at least simulating their movement in the environment. Animation oriented solutions exist, such as crowd patches [6] that define areas with pre-generated animations which can then be connected to create a large environment. Nevertheless, this kind of solution does not fit to game applications for two reasons. First, this method is too restrictive concerning the manageable environments which must fit to the input patches. Second, the crowd cannot be controlled in real-time to handle users' interactions, for example to adapt the autonomous agents' trajectories in order to avoid the player's agent. Another method, combining simulation and graphical *LOD*s [7], allows for more crowd reactivity but still

constrain the behaviours variety by pre-computing possible paths between a limited set of starting / destination points.

To handle the need for highly reactive and varied crowds, the behaviours can be managed using a microscopic simulation model where each agent is endowed with its own decision abilities [8]. The decision can be divided in two main categories. First, the high level behaviours manage the agent's mid term goals and select its immediate action by using a script, a set of rules, or even decision networks [9]. An optimisation of this decision layer, called *LOD-AI* (for LOD Artificial Intelligence), consists in limiting the decision process for the agents which are far from the camera [10]. Second, the low level behaviours manage the agent's ability to move thanks to two complementary mechanisms: the path planning which globally evaluates the movement through the static environment to reach a defined target; and the collision avoidance which locally adjusts the movement in order to avoid dynamic obstacles while following the previously computed path. When considering large crowds of people, both of these mechanisms are computationally critical. The path planning, which highly depends on the environment's scale, can be improved by using a hierarchical evaluation [11]. The collision avoidance is generally evaluated in games by fast rule-based algorithms [12, 13] that allow simulating very large crowds but with poorly convincing behavioural results: wall crossing or late adaptation, mainly because of the difficulty to simultaneously handle multiple potential collisions with static and dynamic entities. A geometric approach [14] enhances the produced trajectory realism at the expense of the computation time, by applying cognitive principles, such as anticipation, while considering all the obstacles at the same time. The particle based approach [15] is more used for high density simulations and is less adapted to animation because of the oscillations in the produced movement and the lack of anticipation in the decision.

We propose in this paper a novel approach focusing on the improvement of the collision avoidance computation performance and realism. The proposed model takes place as a specialised *LOD-AI*, called Collision Avoidance LOD (*CA-LOD*), and is integrated to a modular autonomous agent architecture. We then define three *CA-LOD* represented by a special level without collision avoidance, and two levels of increasing complexity and realism.

2 Overall Approach

2.1 The Simulation Model

We propose to simulate crowds of people using a microscopic simulation model. Thereby, each virtual human composing the crowd is represented by an autonomous agent endowed with individual decision abilities. As shown in *Figure 1*, the first decision mechanism represents *high level behaviours* which select the agent's destination and desired speed. This decision layer will be left apart as it is out of the scope of this paper.

The next mechanism is the *path planning*, that analyses the environment's topology in order to produce and update the path to the selected destination. The environment description we use is based on a Delaunay's triangulation [16], thus the path planning produces a set of edges to traverse in order to reach the destination (see *Figure 2(a)*).

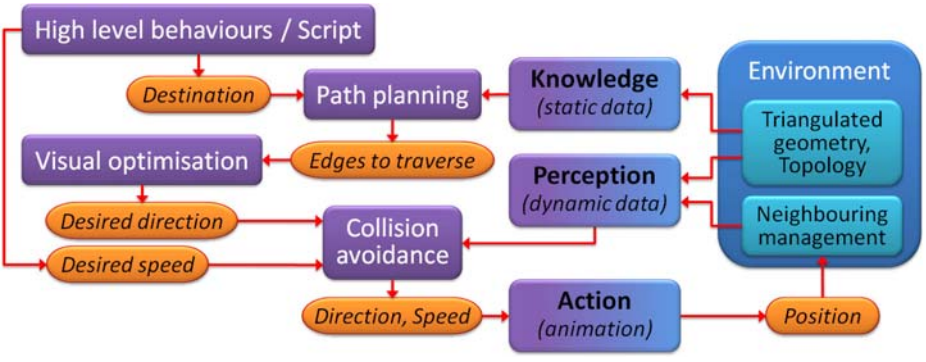


Fig. 1. Virtual human's simulation architecture. In purple, the agent's successive decision mechanisms; in orange, the transferred data.

Then, a complementary process, called *visual optimisation*, analyses the produced path to extract the optimal direction of movement, resulting in an overall path smoothing as shown in *Figure 2(b)*.The triangulated path given to each character as its primary navigation instruction handles all of the static obstacle avoidance, and in general the animated characters follow this visually optimised path directly. Because the path is directly derived from the *geometry* it does not take dynamic obstacles (other moving characters) into account. We therefore need to augment this system with an obstacle-avoidance module.

Our agent architecture is analogous to the classic robot *stack architecture* [17]. As can be seen in *Figure 1*, the collision-avoidance module needs to operate as a closed system that, in only those special conditions where collision-avoidance is necessary, overrides movement instructions from the *visual optimisation* layer. The entities to avoid, i.e. the walls and other agents, are retrieved using a perception algorithm that dynamically extracts the visible elements depending on the agent's direction and field of view

(a) Edges to traverse. (b) Resulting visual optimisation.

Fig. 2. Path planning in Trinity College Dublin. The underlying subdivision is represented with traversable gray edges and obstacle red edges; the resulting path is shown in blue.

as well as the surrounding topology. Finally, the collision avoidance mechanism provides the final direction and speed of movement that are used to animate the agent and to compute its new position.

To conclude, we propose an autonomous agent model managing the human's ability to move. This model allows individual decision, thereby taking into account individual factors such as the destination, the desired speed, or even different preferences for path planning [11]. Moreover, the modularity of this model, with well identified communication between several independent mechanisms, allows one component algorithm to be changed while keeping the entire architecture functional.

2.2 Obtaining a Scalable Model

In order to simulate very large crowds it is essential to achieve the best computation performances. Because the collision avoidance algorithm is highly dependent on external conditions, it needs to be refreshed with a relatively high frequency compared to the other behaviours. This fact, combined with the cost of the algorithm itself, results in a computation bottleneck for large crowds simulation.

The modular nature of our agent architecture gives us some flexibility in terms of the algorithm used for collision-avoidance behaviour. We propose three different collision avoidance models, assigned to three successive levels of detail of increasing computation performance and decreasing realism (*Figure 3(a)*):

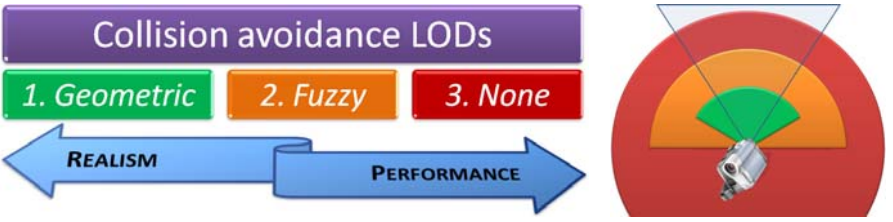
LOD 1 (high). Our geometric model takes into account the entire entity's perception of its immediate surrounding by computing an intermediate geometric representation. The resulting movement is time consistent as other entities' moves are anticipated, providing a high level of realism.

LOD 2 (medium). Our fuzzy logic model only takes into account one potential obstacle at a time. It uses a rule table to quickly make decisions and smoothly interpolate movement instructions. The resulting movement is collision free for low densities of people, but can result in interpenetration for higher densities, providing a restricted level of realism.

LOD 3 (low). The lowest level of detail removes the collision avoidance. This method simply applies the input speed and direction, resulting on an exact path following with agents inter-penetrations, providing a poor level of realism.

In the same way as for geometric LOD algorithms, the Collision Avoidance Level Of Detail (*CA-LOD*) can be chosen depending on the distance to the camera (*Figure 3(b)*): the larger the distance to the camera, the lower the *CA-LOD*; the same principle may be applied for angular distance to the camera direction.

With *CA-LOD*s we propose a novel crowd simulation model, allowing very large crowds to be simulated while keeping precise control over the simulated agents. In the following sections, we detail the collision avoidance models needed for the two higher *CA-LOD*s outlined above, and we present a preliminary comparative benchmarking of these systems.



(a) The three LODs composing the collision avoidance model.

(b) Possible LOD assignment around the camera.

Fig. 3. Scalable collision avoidance principle. In (b) the triangle shows the camera field of view.

3 Fuzzy Steering

We have made use of fuzzy controllers extensively in our previous work [18, 19]. Fuzzy logic has been employed by other agent simulations for its flexible nature - easily expanding to accommodate more complex input variables [20]. Our fuzzy controller in the Metropolis traffic simulation is a two-component system comprising a route-following controller and a dynamic obstacle avoidance controller; these systems are counterbalanced and roughly follow a route of way-points whilst simultaneously avoiding static and dynamic obstacles. To adapt this for our crowd-navigation system, we are using the obstacle avoidance component to only handle special cases where dynamic obstacle avoidance is required. Fuzzy logic allows us to represent a *partial truth*, or imprecise values between *completely true* and *completely false*. This gives us a mechanism for discriminating imprecise or changing data into a small group of overlapping *fuzzy sets*. From this foundation we can create very simple judgement-based reasoning, or *fuzzy inference*, to deal with complex real-world data; mimicking human decision-making. Fuzzy Logic systems require very little computational overhead, and can also produce smooth transitional outputs. Fuzzy Logic is therefore an ideal candidate for modelling human behaviour in large-scale, real time simulations.

In *Figure 4* we consider an example scenario where an agent controls a character that is facing up the page. There are two obstacles nearby; *Obs.0* and *Obs.1*. We have classified the environment into three overlapping sets for distance (*NEA, MED*, and *FAR*), and angle from current heading (*NAR, MID*, and *WID*). Obs.0 is outside our maximum range of consideration, and is thus ignored, Obs.1 is our *nearest obstacle* and straddles both NEA and MED distances, and is in-between both MID and WID angles. In this case we evaluate all four combinations of rules, and blend the results together - weighted by the degree of membership in each of the distance and angle sets. This blending or *aggregation* allows us to smoothly transition between each rule and provides us with very smooth curves of motion that then require no motion post-processing before animations can be applied.

The obstacle-avoidance system considers the *change in heading angle* and *distance* to the nearest obstacle. Both of these systems classify the real inputs into overlapping *rough sets*, where they can be classified in human terms for a quick look-up-table type decision. The mapping of these inputs is designed so that a series of rules will progressively move the subject character away from its route, avoid the obstacle, then finally

return to its route. Figure 4 illustrates this scenario, and shows the range of distances and angles used as input. Table 1 provides our design for fuzzy input set mappings. We produce fuzzy sets for classifying (or *fuzzifying*) real inputs into our rough sets, which are illustrated in Figure 5.

Because we want to represent a range of different characters with the same fuzzy system, we have expressed our outputs in terms of a characters's *maximum velocity* (v_{max}). The steering adjustment outputs are also expressed relative to the characters's current speed; such that resultant movement vector of the character is to some extent scalable, and we can expect similar rates of turn at different speeds. Fuzzy Output set mapping functions for speed and steering are illustrated in Figures 6 *(a)* and *(b)*, respectively.

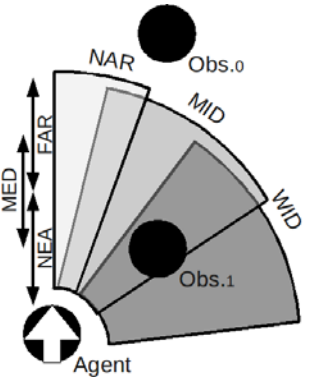


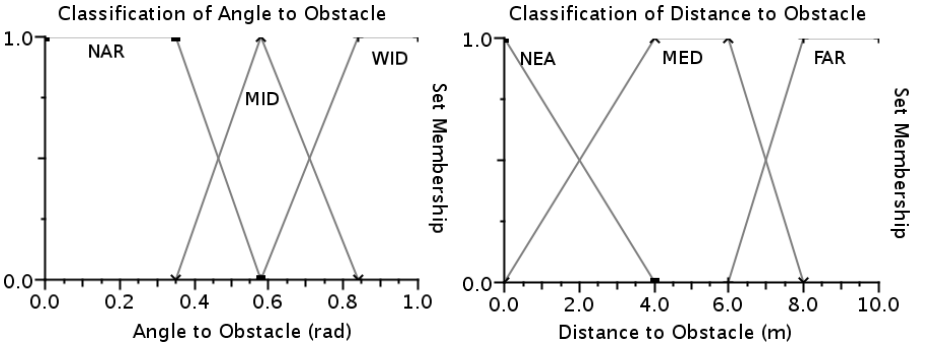
Fig. 4. A scenario illustrating the design of input distances and angles to the fuzzy obstacle avoidance system

The labour-intensive task is then to design fuzzy rules. These map all of the different combinations of fuzzy input values to fuzzy outputs. Whilst we can manually tune these rules to produce a satisfying output for one particular character, the fuzzy systems need to be recalibrated for every character with different performance characteristics or physical dimensions. As we wish to simulate a large variety of characters, this task becomes a serious constraint-based problem which we are tackling in our ongoing works by way of an automatic-calibration and self-training system [21,22]. Our initial rules are provided in Tables 2(a)-2(b). We note that not all of our output sets have been used in these rules, and that there is certainly scope for designing or evolving more balanced rule-sets.

Once all of the rules have been evaluated we use an aggregation *centre-of-gravity* function to merge the outputs for each system. Equation 1 gives us the aggregation function for velocity. To obtain the real output *defuzzified* velocity (v_{defuzz}) we take the membership values (mem_i) for each fuzzy output set (from *ZER* to *TOP*) that we evaluated with our FAMMs, multiply each by its set centre value (v_i) (from Figure 6),

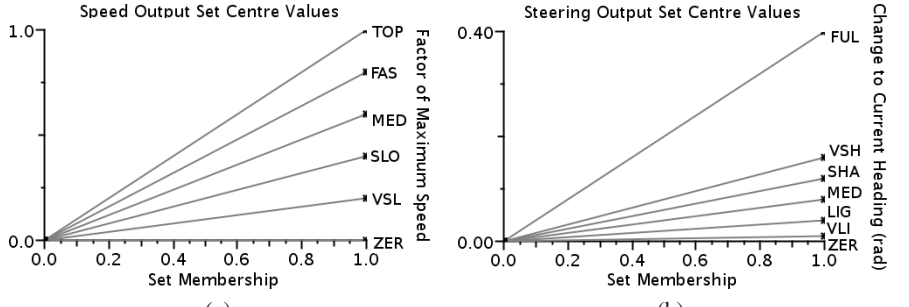
Table 1. Fuzzy Input Term Definitions for obstacle avoidance. An optimisation here is that we are only considering angles from the current heading of a character, on *one side*. We simply flip the same fuzzy sets over to evaluate obstacles on the other side of the character - this then requires half as many fuzzy sets and simplifies calculation.

Real Term	Rough Set Value Range	Fuzzy Term
Wide Arc	$> 0.58rad$	WID
Mid-range Arc	$0.35 - 0.84rad$	MID
Narrow Arc	$0 - 0.58rad$	NAR
Far Distance	$> 6m$	FAR
Medium Distance	$0 - 8m$	MED
Near Distance	$0 - 4m$	NEA



(a) (b)

Fig. 5. Fuzzy Input Set Membership Functions for classifying the *angle (a)* and *distance (b)* to the nearest obstacle in fuzzy terms. Angles here are absolute radians to the left or right of the current heading of a character, so that obstacles on the left hand side of a character are treated the same as obstacles to the right.



(a) (b)

Fig. 6. Fuzzy Output Value centres for obstacle avoidance desired *speed* as a factor of maximum speed and *steering adjustment*; a modifier. The defuzzified output speed factor will be multiplied with the character's top allowable speed to produce a desired speed as a crisp number. The steering modifier is subtracted from the route-following steering factor.

Table 2. 3x3 Fuzzy Associative Memory Matrices (FAMMs) for Desired Speed (a) and Steering (b). These tables are the core of the fuzzy system, and express the rules that match each possible combination of fuzzy input angles and distances to a fuzzy output. We can see that if an obstacle is a *near* distance at a *mid-range* angle then we will travel at a *very slow* speed and make a *medium* turn away from the obstacle. This system provides a very fast rule look-up table for real-time operation.

(a) Desired Speed.

	NEA	MED	FAR
NAR	ZER	VSL	SLO
MID	VSL	SLO	MED
WID	SLO	MED	FAS

(b) Desired Steering.

	NEA	MED	FAR
NAR	SHA	MED	VLI
MID	MED	VLI	ZER
WID	VLI	ZER	ZER

sum the result together, and finally divide this by the sum of all of the centre values.

$$v_{defuzz} = \sum_{i=top}^{zer} (mem_i.v_i) \; / \; \sum_{i=top}^{zer} v_i \tag{1}$$

4 Geometric Steering

The geometric steering model avoids collisions based on the following assumptions:

- Resolving collision avoidance consists of selecting the instantaneous speed and direction of movement;
- The selected movement must take into account the static and dynamic obstacles, as well as the desired movement of the entity;
- The selected movement must be as consistent as possible through time, avoiding decision oscillations.

The proposed model is similar to the potential fields because it merges multiple collisions sources in a single representation. Nevertheless, our model adds two major advantages by allowing individual reasoning and subdividing the environment in a more accurate way than with grids (which are commonly used for potential fields). Three successive steps are necessary to take the optimal movement decision:

1. Construct a local geometric representation of the agent's surroundings, merging information about the static and dynamic obstacles while computing area potentials;
2. Extract from the previous representation the area with the best potential, considering the environmental constraints and the agent's desired movement;
3. Constrain the agent's movement by the movement allowed in the previously selected area.

Even if this approach is based on our previous work [14] we have improved this technique, notably considering the way to construct the temporary representation. Let us now detail each computation step.

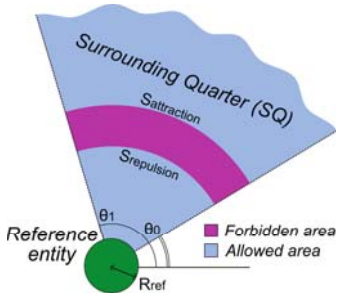
4.1 The Local Geometric Representation

The local geometric representation subdivides the agent's surrounding in quarters storing the movement's constraints for a direction interval (*Figure 7*(a)). The following data are stored for each surrounding quarter SQ:

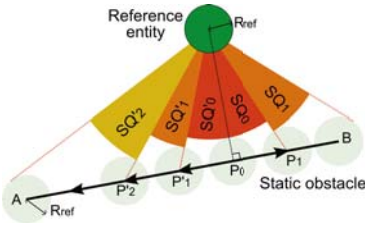
The angle interval. $[\theta_0;\theta_1]$ defining the range of directions covered by the quarter.

The time interval. t defining the time range $[R_t;R_{t+1}]$ with $R_t = \begin{cases} 0 & \text{if } t = 0 \\ k^t & \text{otherwise} \end{cases}$ corresponding to the quarter data; the time range follows an exponential scale to obtain a more precise subdivision in the near future while allowing long time prediction (we use $k = 2$ in our simulations).

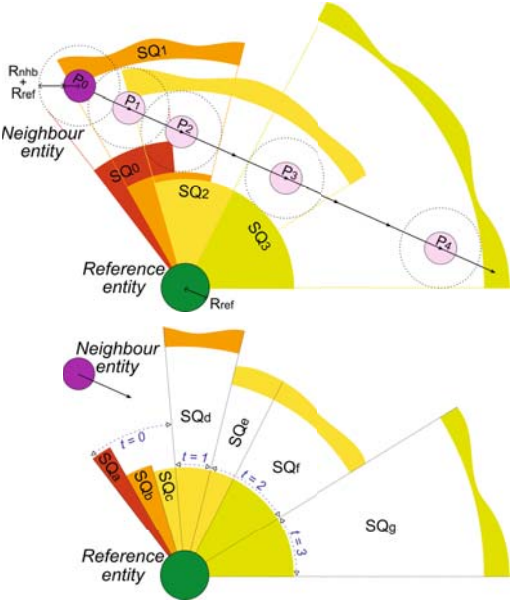
The repulsion speed. $Sr \geq 0$ defining the maximal allowed speed to avoid collisions.



(a) Surrounding quarter principle.



(b) Static obstacle subdivision.



(c) Dynamic obstacle subdivision: on top, per time interval quarter extraction; on bottom, resulting intersected quarters.

Fig. 7. Local representation used for the geometric collision avoidance. For simplification purpose, the repulsive and attractive distances are represented instead of the corresponding speeds.

The attraction speed. $Sa \geq 0 \in$ defining the minimal necessary speed to avoid collisions.

The potential factor. F which scores the quarter's attractiveness for the agent's next move (the lower the best); we will see later that this factor reflects the agent's *effort* to avoid a collision in this quarter.

Initially, only one quarter exists with the following values: $[\theta_0;\theta_1] = [0;2\pi]$ which covers the entire agent's surrounding; $Sr = Sa = undefined$ which is a special value meaning that the quarter does not constrain the agent's speed; $F = 0$ which is the neutral potential value; and $t = undefined$. When a new quarter SQ_n is inserted, the existing overlapped quarters SQ_o are intersected to produce the appropriate subdivision, and their data are updated based on these rules:

- **if** $\begin{pmatrix} \vee & t_o > t_n & \wedge\ Sr_o = undefined \\ \vee & (\ t_o = t_n \vee t_o = undefined \vee t_n = undefined\ ) & \wedge\ Sr_n \neq undefined \\ & & \wedge\ Sr_n < Sr_o \end{pmatrix}$ **then** $Sr_o \leftarrow Sr_n$
- **if** $\begin{pmatrix} \vee & t_o > t_n & \wedge\ Sa_o = undefined \\ \vee & (\ t_o = t_n \vee t_o = undefined \vee t_n = undefined\ ) & \wedge\ Sa_n \neq undefined \\ & & \wedge\ Sa_n > Sa_o \end{pmatrix}$ **then** $Sa_o \leftarrow Sa_n$
- **if** $\left(t_n \neq undefined \wedge t_n < t_o\right)$ **then** $t_o \leftarrow t_n$
- $F_o \leftarrow F_o + F_n$

Static obstacle subdivision. We propose two subdivision methods, one for the static obstacles (typically walls) and the other for the dynamic obstacles (other agents). The static obstacles are subdivided in quarters around the projected point P_0 of the agent's position (*Figure 7*(b)). The subdivision follows an exponential step on the same way as the exponential time intervals. Defining $[AB]$ the obstacle segment and S_{ref} the agent's desired speed, each subdivision point P_i is obtained by the following formula:

$$\begin{array}{l} P_i = P_0 + \vec{\Delta_i} \\ P'_i = P_0 - \vec{\Delta_i} \end{array} \text{ where } \vec{\Delta_i} = k^{i-1}.\frac{S_{ref}}{\| \vec{AB} \|}.\vec{AB} \text{ for } i > 0$$

The successive pairs of points $(P_i; P_{i+1})$ are used to compute each quarter SQ_i; if both points are not on the segment $[AB]$ they are ignored; if one point is not on $[AB]$, the corresponding quarter's angle is replaced by the external tangent to a circle with the same radius as the agent and centred on the corresponding segment's extremity. Finally, each quarter's internal data are computed on this way:

- $t = \textit{undefined}$ because neither quarter corresponds to predicted data, but only to a geometric subdivision.
- $Sr_i = \| \vec{P_{ref}P_i} \| - R_{ref}$ with P_{ref} the position of the reference agent.
- $Sa = \infty$ because the obstacle cannot be crossed.
- $F_i = \begin{cases} 0 & \text{if } S_{ref} \leq Sr_i \\ 1 - Sr_i/S_{ref} & \text{otherwise} \end{cases}$

Dynamic obstacle subdivision. The dynamic obstacles' quarters are obtained by anticipating the movement of a neighbour considering his current velocity $\vec{V_{nhb}}$ (top of *Figure 7*(c)). A predicted position P_t is computed for each time value $t > 0$ by $P_t = P_0 + k^{t-1}.\vec{V_{nhb}}$. Then, each pair of successive position $(P_t; P_{t+1})$ is used to configure a quarter SQ_t: the angles are the outside tangents from the reference agent's position P_{ref} to the circles defined by P_t and P_{t+1} with a radius $R_{sum} = R_{nhb} + R_{ref}$. Each quarter's internal data are computed on this way, assuming $Dmin_t = \min\{\| \vec{P_{ref}P_t} \|; \| \vec{P_{ref}P_{t+1}} \|\}$ and $Dmax_t = \max\{\| \vec{P_{ref}P_t} \|; \| \vec{P_{ref}P_{t+1}} \|\}$:

- t is the value used to compute the quarter.
- $Sr_t = (Dmin_t - R_{sum})/k^t$
- $Sa_t = \begin{cases} \infty & \text{if } t = 0 \text{ (neighbour's current position is uncrossable)} \\ (Dmax_t + R_{sum})/k^{t-1} & \text{otherwise} \end{cases}$
- $F_t = \begin{cases} 0 & \text{if } S_{ref} \leq Sr_t \vee S_{ref} \geq Sa_t \\ \min\{S_{ref} - Sr_t; Sa_t - S_{ref}\}/(S_{ref}.k^t) & \text{otherwise} \end{cases}$

4.2 Selecting the Best Movement Area

Once all the surrounding obstacle have been processed and merged into the geometrical representation, the best movement area can be obtained by selecting the quarter with the lowest potential factor. In order to take into account the agent's desired movement

direction θ_{ref}, an additional direction change factor F_d is computed. The relative importance $\lambda \geq 0$ of the direction change toward the speed change is then applied in order to obtain the finally compared factor $F_f = F_q + \lambda.F_d$

The direction-change factor reflects the agent's effort to deviate from his optimal trajectory (*Figure 8*). It is sensible to a maximal allowed direction change $0 < \theta_{max} \leq \pi$, which can be configured depending on the agent's physical constraints for example. Defining θ_{SQ} the angular distance between a quarter and D_{ref} (which may be null if the quarter encloses D_{ref}), the direction-change factor is computed this way:

$$F_d = \begin{cases} \infty & \text{if } \theta_{SQ} > \theta_{max} \\ \theta_{SQ}/\theta_{max} & \text{otherwise} \end{cases}$$

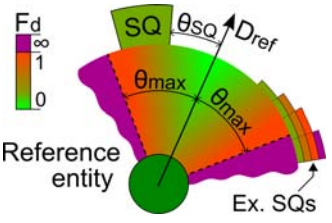


Fig. 8. Direction-change factor computation

4.3 Final Movement Decision

When the best surrounding quarter is selected, obtaining the final movement is trivial. The collision free direction is the nearest direction to D_{ref} that is covered by the quarter. The final speed S_{fin} is computed based on the selected quarter's data, thanks to this algorithm:

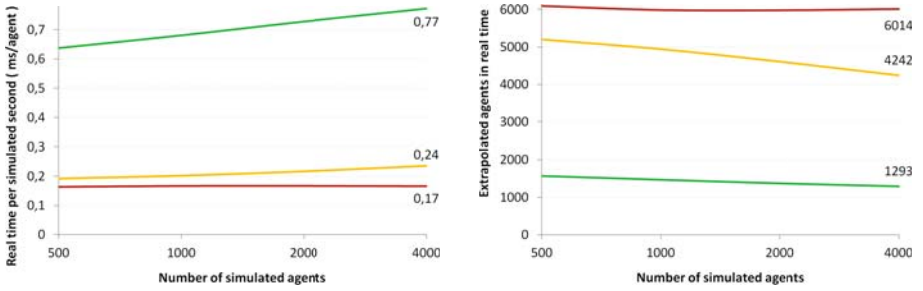
```
1  if S_ref ≤ Sr ∨ S_ref ≥ Sa then
2      S_fin ← S_ref
   else
3      Dec ← S_ref − Sr
4      Acc ← Sa − S_ref
5      if Dec ≤ Acc then
6          S_fin ← Sr
       else
7          S_fin ← Sa
       end
   end
```

Lines $1-2$ manage the case where the agent's movement is not constrained, allowing the desired speed. In the other case, the necessary deceleration (l.3) or acceleration (l.4) are computed. The less constraining option is then chosen (l.5), resulting in a slower (l.6) or faster (l.7) speed.

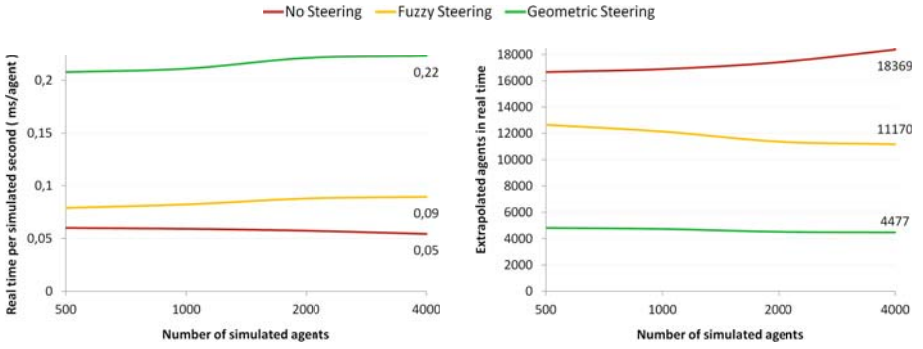
5 Results

The proposed virtual human model has been implemented into a crowd animation tool called *Metropolis*. This tool is used to animate huge crowds of people, with a final objective to simulate Dublin city in real time with highly capable virtual humans. Each collision avoidance model has been tested in Metropolis, but the LOD selection algorithm is still a work in progress.

We have run series of tests with each collision avoidance model to obtain their respective performances, allowing us to have an indication of the benefit to mix these models in the final *CA-LOD* framework. These experiments only take into account the behavioural computation, the rendering being deactivated to not influence the results.



(a) Single thread performances of the collision avoidance algorithms.



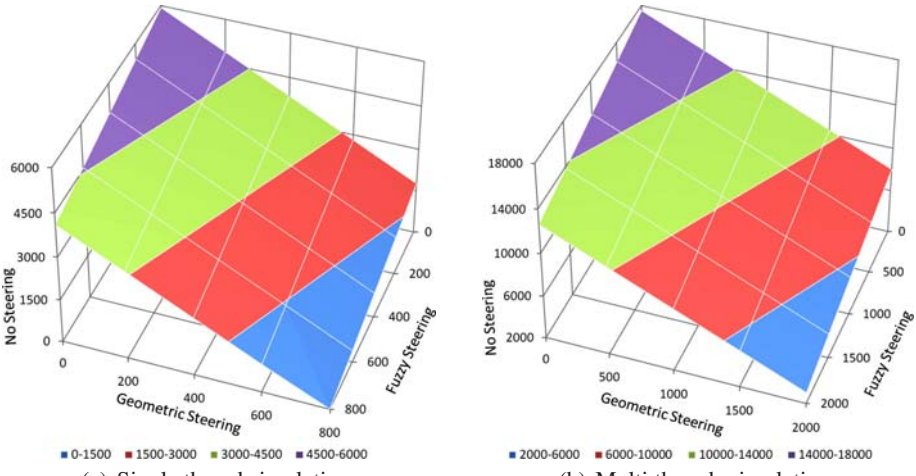
(b) Multi threads performances of the collision avoidance algorithms.

Fig. 9. Performances of each collision avoidance algorithm

Each agent is randomly generated in a navigable area of the Trinity College virtual environment (approximately $2km^2$), and pick a random target in the same way (which is renewed when reached).

Figure 9 shows the performances obtained on an Intel Core2 Quad Q8200 2.33GHz. These results are an average of four simulation runs during fifteen simulated minutes. For convenience, each test is represented on the left with the required real time to compute an agent during one simulated second, and on the right with the corresponding expected simulated agents in real time. These graphs show that the no-steering model has a relatively constant computation performance whatever the number of simulated agents, because it does not take into account the agent's neighbourhood. This model displays the maximal computation performance we can expect from the *CA-LOD* model. The fuzzy-steering model is approximately 35% slower, with a small influence of the total number of simulated agents, mainly because this model only takes into account one near neighbour at a time. The geometric-steering model is 75% slower than without steering, with also a small influence of the overall number of simulated agents because part of the input is the topology which is unchanging.

Finally, an extrapolation of the possible repartition between the models to obtain real time performances is shown in *Figure 10*. The expected performances are very good, allowing thousands of agents to be simulated even in single thread with hundreds of agents using the best *CA-LOD*.



(a) Single thread simulation. (b) Multi threads simulation.

Fig. 10. Possible models repartition, in number of agents, allowing real-time performances

6 Conclusion and Perspectives

We have presented in this paper a novel framework for autonomous agents involving Collision Avoidance Levels Of Detail. Our objective is to propose a highly scalable model while maintaining the simulated crowd controllable to allow interactive applications. Three *CA-LOD*s of increasing complexity are proposed and described: the deactivation of the steering behaviour for the worst case; a fuzzy steering only taking into account one obstacle for the medium case; and a geometric steering with anticipation over all surrounding obstacles for the best case. An evaluation of the computation cost for each collision avoidance model is finally done, from which we extrapolate the possible repartitions between the models in order to obtain real-time performances on a reference computer. The obtained results allow to foresee real-time applications with huge interactive crowds.

Future work will first focus on the evaluation of different strategies allowing to choose each agent's *CA-LOD*. Indeed, we want to propose a model capable of choosing the optimal repartition between the *CA-LOD*s in order to maintain real-time performances while achieving the best visual realism. To do so, perceptual studies will be done in order to evaluate the areas where each collision avoidance model can be used to obtain the minimal realism loss. Then, an algorithm will be proposed to automatically choose the best repartition based on the dynamic analysis of the simulation performances. We envisage to use a genetic algorithm in order to automatically configure the best repartition based on measurable criteria, such as the number of non-avoided visible collisions with respect to the camera distance. Another perspective concerns the extension of the number of managed *CA-LOD*s, for example by limiting the inputs of the geometric steering to a maximal number of neighbouring obstacles. Finally, we plan to extend this approach to handle on the same decision basis all the possible level of details for an autonomous agent, such as other *LOD-AI*s, animation *LOD*s, or geometric *LOD*s.

Acknowledgement

We wish to thank *Science Foundation Ireland* (SFI) for funding the *Metropolis* project. We also want to thank Simon Dobbyn for his great work on the *Metropolis* system, as well as the entire *GV2* team for their participation to all parts of this project.

References

1. Hamill, J., O'Sullivan, C.: Virtual dublin - a framework for real-time urban simulation. In: Proc. of the Winter Conference on Computer Graphics, vol. 11, pp. 1–3 (2003)
2. Peters, C., Ennis, C.: Modeling groups of plausible virtual pedestrians. IEEE Computer Graphics and Applications 29(4), 54–63 (2009)
3. Wimmer, M., Bittner, J.: Hardware occlusion queries made useful. In: Pharr, M., Fernando, R. (eds.) GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation. Addison-Wesley, Reading (2005)
4. Luebke, D., Watson, B., Cohen, J.D., Reddy, M., Varshney, A.: Level of Detail for 3D Graphics. Elsevier Science Inc., New York (2002)
5. Dobbyn, S., Hamill, J., O'Conor, K., O'Sullivan, C.: Geopostors: a real-time geometry/impostor crowd rendering system. ACM Trans. Graph. 24(3), 933 (2005)
6. Yersin, B., Maim, J., Pettré, J., Thalmann, D.: Crowd Patches: Populating Large-Scale Virtual Environments for Real-Time Applications. In: I3D 2009 (2009)
7. Pettré, J., de Ciechomski, P.H., Maïm, J., Yersin, B., Laumond, J.P., Thalmann, D.: Real-time navigating crowds: scalable simulation and rendering. Computer Animation and Virtual Worlds 17(3-4), 445–455 (2006)
8. Paris, S., Donikian, S.: Activity-driven populace: a cognitive approach for crowd simulation. Computer Graphics and Applications (CGA) special issue Virtual Populace 29(4), 24–33 (2009)
9. Yu, Q., Terzopoulos, D.: A decision network framework for the behavioral animation of virtual humans. In: Metaxas, D., Popovic, J. (eds.) Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, pp. 119–128 (2007)
10. O'Sullivan, C., Cassell, J., Vilhjálmsson, H., Dingliana, J., Dobbyn, S., McNamee, B., Peters, C., Giang, T.: Levels of detail for crowds and groups. Computer Graphics Forum 21(4), 733–741 (2003)
11. Paris, S., Donikian, S., Bonvalet, N.: Environmental abstraction and path planning techniques for realistic crowd simulation. Computer Animation and Virtual Worlds 17, 325–335 (2006)
12. Reynolds, C.W.: Steering behaviors for autonomous characters. In: Game Developers Conference 1999 (1999)
13. Lamarche, F., Donikian, S.: Crowds of virtual humans: a new approach for real time navigation in complex and structured environments. Computer Graphics Forum 23, 509–518 (2004)
14. Paris, S., Pettré, J., Donikian, S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. In: Computer Graphics Forum, Eurographics 2007, vol. 26(3), pp. 665–674 (2007)
15. Helbing, D., Buzna, L., Johansson, A., Werner, T.: Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. Transportation Science 39(1), 1–24 (2005)
16. Paris, S., Mekni, M., Moulin, B.: Informed virtual geographic environments: an accurate topological approach. In: The International Conference on Advanced Geographic Information Systems & Web Services (GEOWS). IEEE Computer Society Press, Los Alamitos (2009)

17. Choset, H.M., Hutchinson, S., Lynch, K.M., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementation. MIT Press, Cambridge (2005)
18. Gerdelan, A.P.: A solution for streamlining intelligent agent-based traffic into 3d simulations and games. Technical Report CSTN-072, IIMS, Massey University, North Shore 102-904, Auckland, New Zealand (January 2009)
19. Gerdelan, A.P.: Driving intelligence: A new architecture and novel hybrid algorithm for next-generation urban traffic simulation. Technical Report CSTN-079, Institute of Information and Mathematical Sciences, Massey University, North Shore 102-904, Auckland, New Zealand (February 2009)
20. Dougherty, M., Fox, K., Cullip, M., Boero, M.: Technological advances that impact on microsimulation modelling. Transport Reviews 20(2), 145–171 (2000)
21. Gerdelan, A.P., Reyes, N.H.: Towards a generalised hybrid path-planning and motion control system with auto-calibration for animated characters in 3d environments. In: Advances in Neuro-Information Processing. LNCS, vol. 5507, pp. 25–28. Springer, Heidelberg (2008)
22. Gerdelan, A.P.: Architecture design for self-training intelligent vehicle-driving agents: paradigms and tools. Technical Report CSTN-088, Institute of Information and Mathematical Sciences, Massey University, North Shore 102-904, Auckland, New Zealand (April 2009)