EACS: Effective Avoidance Combination Strategy

Julien Bruneau, Julien Pettré

Volume xx (200y), Number z, pp. 1–12

EACS: Effective Avoidance Combination Strategy

J. Bruneau[†1] and J. Pettré[‡1]

[1]Inria Rennes, France

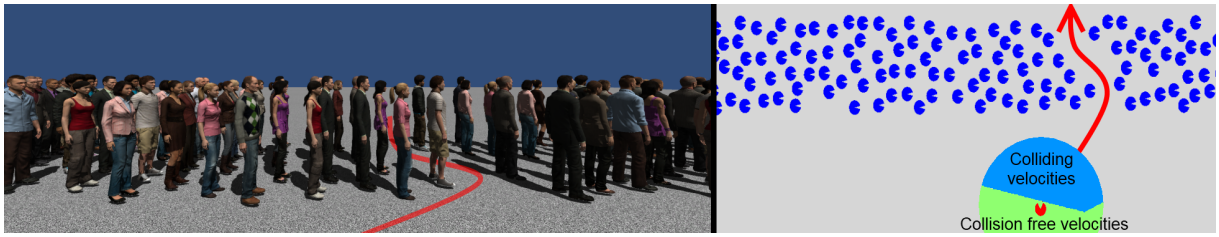

Figure 1: *In some situations, humans can see a tunnel through a flow. The left figure shows a first person view point where the tunnel is clearly visible. The right figure shows a top view of the situation as well as the set of colliding and collision-free velocities for the red agent. Regular local avoidance models try to find a collision-free speed toward the goal. This means the red agent will choose a velocity from the set of collision-free velocities (green part of the circle), making him slow down and miss the tunnel.*

Abstract

When navigating in crowds, humans are able to move efficiently between people. They look ahead to know which path would reduce the complexity of their interactions with others. Current navigation systems for virtual agents consider long-term planning to find a path in the static environment and short term reactions to avoid collisions with close obstacles. Recently some mid-term considerations have been added to avoid high density areas. However, there is no mid-term planning among static and dynamic obstacles that would enable the agent to look ahead and avoid difficult paths or find easy ones as humans do. In this paper we present a system for such mid-term planning. This system is added to the navigation process between pathfinding and local avoidance to improve the navigation of virtual agents. We show the capacities of such a system using several case studies. Finally we use an energy criterion to compare trajectories computed with and without the mid-term planning.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.5 [Simulation and Modeling]: Types of Simulation—Animation;

1. Introduction

Crowd simulation is a very active field with many applications, such as the entertainment industry, with the aim of populating scenes for games and movies. For these applications we need to create virtual humans (agents) that behave in a believable manner. Spectators easily detect any strange behaviors, such as unnatural navigation trajectories. The main objective of this paper is to improve agents' navigation skills. In particular, we reinforce the capacity of the agents to setup relevant mid-term strategies for navigation among moving obstacles (typically, a crowd of moving agents).

A typical problem in agent navigation which remains quite neglected is how agents combine successive interactions in time. Today, most crowd simulators handle navigation in two steps. First, the long term strategy: the agent

† e-mail: julien.bruneau@irisa.fr

‡ e-mail: julien.pettre@inria.fr

searches for a path between its position and its goal in the environment. This step is done by pathfinding modules. The second step is a short term strategy: while following the path the agent adapts its velocity to avoid any collision with nearby obstacles. This step is done by local avoidance modules. Most advanced solutions are able to combine simultaneous interactions: agents will find velocities that avoid collisions with all its neighbor obstacles at the same time. But such velocities do not always exist, especially when the agent is surrounded by many obstacles. In such a situation, local avoidance models have to relax their constraints and allow colliding velocities. Moreover, the consequences of one local avoidance strategy (i.e., nature of the adaptations made) on the following interactions are generally not considered. Our objective is to prevent agents from performing an unnatural sequence of successive avoidance motions which could get them stuck, if there are some more relevant and obvious solutions.

Real humans are very efficient at snaking their way through dense crowds. An example is shown in Figure 1, where a red agent needs to cross a dense flow. There is an empty space in this flow that we call a tunnel. If we look at the first person view point on the left of the figure, the tunnel is clearly visible. A real human would probably detect it and use it to cross the flow with minimum effort. Pathfinding techniques are unable to detect this tunnel as they plan inside the static environment and do not have information about the dynamic obstacles around them (i.e., other agents). Local avoidance systems are also unable to detect this tunnel because they consider short term strategies only; they try to avoid collisions with only one velocity adaptation. The collision-free velocity space and the colliding velocity space, shown on the figure, have been computed for a near future as done in local avoidance models. A closer look at the collision-free velocities (in green) shows that the agent can either turn right and follow a parallel path to the flow forever or turn left/reduce speed and pass behind the flow. But there is no indication of a tunnel in the flow on the different velocity spaces used by local avoidance to take a decision. In this situation local avoidance models make the red agent go straight forward and slow down a lot when approaching the flow while the blue agents (in the flow) will move to let the red agent pass (see Figure 10). More recently, some heuristics have been proposed to choose a strategy depending on density, but these heuristics would also fail in our example as the tunnel is not straight and it requires considering several velocity adaptations to cross the flow. In this paper, we explore a third step for navigation in crowd simulators, which looks further into the future than the local avoidance step does, but not as far as the path finder does. Such a step is thus between pathfinding and local avoidance. It provides a mid-term strategy, allowing the agent to handle the kind of situations shown in Figure 1.

There exist an infinite number of solutions to avoid a collision with an obstacle. An avoidance strategy can be defined as the nature of adaptations made to the trajectory to perform avoidance. A collision can for example be avoided by adapting the speed or orientation, or a combination of both. The strategy choice has a huge impact on the resulting trajectories since, besides preventing the current collision, it also influences the relative motion with all the other obstacles, and thus changes the way the next potential collisions will be avoided. In this paper, we propose a solution to handle mid-term strategies by building several strategies to navigate through a crowd. These strategies consist of a list of adaptation sequences over time. The less costly strategy is then chosen. The cost of a strategy is a composition of the energy needed to follow the strategy, as well as an extra cost for every collision caused by the strategy. This allows agents to plan their way through a crowd, using empty spaces to navigate while optimizing their effort. We show the improvement compared to local avoidance systems in terms of energy consumption.

Our contribution is a mid-term motion planning technique for complex scenarios with multiple moving obstacles. Our system, called Effective Avoidance Combination Strategy (EACS), is able to compute an energy-efficient avoidance path made of successive adaptations. To this end, it explores several possible ways of combining interactions, evaluates the energy cost of each possible solution, and selects the most efficient one.

The remainder of this paper is organized as follows. In Section 2, we discuss the previous models used in agents' navigation. In Section 3.1, we present our system and explain our approach. Section 4 shows the capabilities of our system on some specific situations, and an evaluation in terms of energy gain with respect to previous models. Finally, in Section 5, we draw some conclusions.

2. Related Work

In a crowd simulation, the navigation of agents generally results from two interlinked components. A global path planner is in charge of providing a sequence of way-points to guide agents to long-term goals. Some planners were specifically designed to handle crowds ([PCM*06], [BLA02], [KGO09], [ST05]). While agents follow the planned motion, the local avoidance system avoids collisions with moving obstacles that were not considered by the global planner ([Rey87], [vdBLM08], [HFV00]). Between these two levels of motion synthesis a relatively important gap exists.

Global path planners were extended to consider the presence of dynamic obstacles: a graph can vary through time to be adapted to dynamic obstacles [SGA*07] or information such as local density can be added to the graph [PCM*06], including a prediction of its evolution over time [KBG*13]. However, these solutions are not able to estimate an optimal strategy over a sequence of interactions. Sud and colleagues [SGA*07] consider agents' positions only and do

not anticipate future agents' trajectories, thus preventing them from exploring relevant strategies for successive interactions in time. Kapadia and colleagues [KBG*13] consider the case of multiple interactions by merging them to penalize going through denser areas and adding time consideration to compute collision-free velocities with neighbors in a similar way as velocity-based local avoidance models. But they are not able to explore avoidance strategies for successive interactions. Our system could actually be used to extend this planning framework with a finer-grained interaction analysis.

Local avoidance systems were also extended to consider some elements of mid-term strategies. Guy and colleagues [GCC*10] estimate the cost of applying a given strategy to perform collision avoidance. However, its effect on the following interactions was explored only latter by Godoy et al. [GKGG14] who 'peek' on potential future scenarios when evaluating different strategies. Golas in [GNCL14] looks for the density of the crowd further ahead and penalizes avoiding velocities that go toward high density areas. Using the same principle, [BNCM14] adapts the walking speed to the crowd density ahead of the agent. These methods allow improvements over agents' navigation, but still limit themselves to a single velocity adaptation which is not sufficient to solve situations such as the one presented in the introduction and shown in Figure 1. The method presented in [KSHF09] uses egocentric affordance fields to find a path through dynamic and static obstacles, and should be able to solve the situation of Figure 1. While the egocentric field should contain enough information on the immediate surroundings of the agent to find such narrow passageways, information is lost over distance and narrow passageways will not be detected if too far away, even if there is no obstruction between the agent and the passageways.

Our approach differs from the previous work in various aspects. The proposed method is based on an accurate evaluation of the cost of several feasible paths; it is not guided by heuristics which only consider the effect of density on navigation efficiency. Our method finds good strategies to combine interactions in time by selecting strategies that may improve several successive avoidance interactions. It differs from techniques which apply a greedy approach by selecting the best strategy for the next interaction only, such as [BNCM14]. The most similar approach to ours is probably the one by Godoy and colleagues [GKGG14], with the major difference that we explore the n next interactions occurring in time, whereas they considers a specific time frame in the future and test several trajectories partitioned in time.

3. Effective Avoidance Combination Strategy

3.1. Overview

The objective of the EACS system (EACS stands for Effective Avoidance Combination Strategy) is to enable agents to perform efficient navigation around other moving agents.

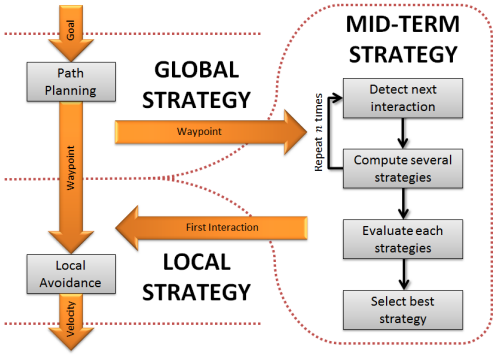

Figure 2: *Navigation system architecture with three different levels of decision.*

The typical structure of a crowd simulation framework consists of two levels. As presented in Figure 2, there is the path planning for global strategy and the local avoidance for local strategy. The path planning finds a path in the environment which consists of a sequence of positions without any fixed obstacles between two consecutive positions. The local avoidance uses the first position of the path, which is reachable in a straight line from the agent's position, and computes a speed to reach it without any collision with nearby fixed or moving obstacles. The EACS system stands between these two modules; it takes the reachable first position of the path from the path planning as a mid-term goal and outputs a preferred velocity to the local avoidance system. The local avoidance system will then follow this preferred velocity while avoiding very close obstacles. Efficient navigation is obtained by setting the relevant avoidance strategy for the few next future interactions with anticipation.

To this end, EACS detects all next interactions on the path to the mid-term goal and computes a sequence of velocity adaptations to successively perform collision avoidance. The variety of possible adaptations to achieve collision-free motion results in different mid-term navigation paths. EACS evaluates and compares their efficiency from the energy point of view and selects the most efficient one to steer agents. In the remainder of this section we start by defining the main concepts used in our system, followed by a description of our technique.

3.2. Definitions

Collision Course: two agents are on a collision course when their position and velocity vectors are set to cause their future distance of closest approach to be below contact distance if the velocity vector is kept constant.

Avoidance Strategy is the nature of the adaptations made to velocity vectors to avoid future collisions. For example, an agent can avoid a collision with another agent by adapting

speed (i.e., the norm of the velocity vector) or by turning (i.e., by rotating the velocity vector).

Interaction Waypoints (IW): for two agents on a collision course and performing collision avoidance, the IW are the space-time waypoints by which agents go when avoiding.

Cardinal Interaction Waypoints (CIW) are the 4 IW which result from 4 specific avoidance strategies with the minimum required amount of adaptation: accelerating only, decelerating only, turning left only and turning right only. We note these cardinal interaction points by **a**, **d**, **l** and **r**, respectively.

Interaction Paths (IP) are a sequence of IW ordered by increasing time. An agent that follows an IP will go through all its IW. Between two consecutive IW the agent follows the straight line between the two positions at a constant speed.

Interaction Segments (IS) are a part of an IP composed of two consecutive IW.

Goals (G) are special IW that do not have a time constraint. A goal is a position in space that agents try to reach, regardless of the when reach it. Goals appear only at the end of an IP or IS. When an agent is following an IS between a regular IW and a goal, it simply walks at its preferred speed toward the goal.

3.3. Cardinal Interaction Points (CIW)

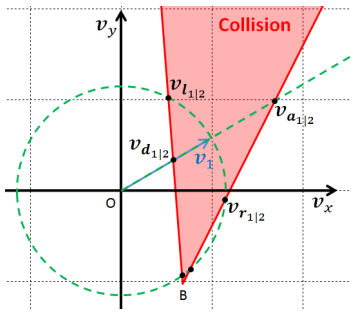

Figure 3: *Collision velocity space for one interaction, in red.* $\mathbf{v}_{\mathbf{a}_{1|2}}$, $\mathbf{v}_{\mathbf{r}_{1|2}}$, $\mathbf{v}_{\mathbf{d}_{1|2}}$ *and* $\mathbf{v}_{\mathbf{l}_{1|2}}$ *represent the collision-free velocities that are used to compute the CIW.*

In this section, we describe how we compute the 4 CIW. We arbitrarily chose these four points covering the four types of adaptation (accelerating, decelerating, turning right and turning left) as well as the two orders of passage (passing before the other agent or after). Using these 4 CIW yields satisfying results, but other IW could be consider when computing avoidance strategies.

The 4 CIW computation is described for a given agent α_1 on a collision course with agent α_2. An example is presented in Figure 4. Their respective 2-dimensional positions and velocities at current time t_0 are noted $(\mathbf{p}_1, \mathbf{v}_1)$ and $(\mathbf{p}_2, \mathbf{v}_2)$. The CIW (3-dimensional: space and time) are noted: $\mathbf{a}_{1|2}$, $\mathbf{d}_{1|2}$, $\mathbf{l}_{1|2}$ and $\mathbf{r}_{1|2}$. To compute these points, we base our principle on the velocity-obstacle introduced in [FS98] as illustrated in Figure 3. The key idea is to compute the 4 admissible velocities which enable collision-free trajectories and which correspond to 4 specific strategies (deceleration, acceleration, left turn, right turn). We recall that the set of admissible velocities for α_1 concerning its interaction with α_2 is noted $AV_{1|2}$ and is defined as follows:

$$AV_{1|2} = \{\mathbf{v}_1 \in V_1 \mid \forall t \in [t_0, t_0+\tau],\ dist_{1,2}(\mathbf{v}_1, t) \geq c\} \quad (1)$$

where: V_1 is the set of all the reachable velocities for α_1, τ is the size of a time window, $dist_{1,2}(\mathbf{v}, t) = \|(\mathbf{p}_2 + \mathbf{v}_2 t) - (\mathbf{p}_1 + \mathbf{v}t)\|$ and c is the collision distance threshold.

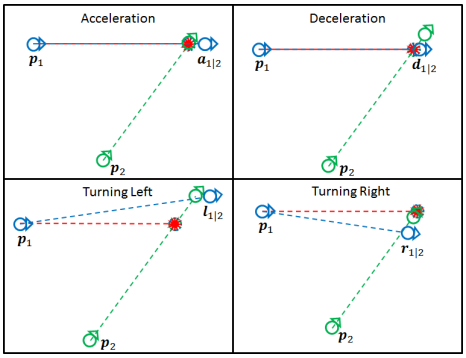

Figure 4: *Example of collision solved using the four CIW. The collision trajectories are drawn in red and the four trajectories followed to reach the CIW are shown in blue. The position of each CIW is shown together with the position of the other agent when the CIW is reached.*

Then, the CIW define trajectories followed by α_1 with specific strategies (velocities): $\mathbf{v}_{\mathbf{a}_{1|2}}, \mathbf{v}_{\mathbf{d}_{1|2}}, \mathbf{v}_{\mathbf{l}_{1|2}}, \mathbf{v}_{\mathbf{r}_{1|2}} \in VA_{1|2}$:

- $\mathbf{v}_1 \cdot \mathbf{v}_{\mathbf{a}_{1|2}} = 0$, $\|\mathbf{v}_1\| < \|\mathbf{v}_{\mathbf{a}_{1|2}}\|$, and $\exists t \in [t_0, t_0+\tau] \mid dist_{1,2}(\mathbf{v}_{\mathbf{a}_{1|2}}, t) = c$,
- $\mathbf{v}_1 \cdot \mathbf{v}_{\mathbf{d}_{1|2}} = 0$, $\|\mathbf{v}_1\| > \|\mathbf{v}_{\mathbf{d}_{1|2}}\|$, and $\exists t \in [t_0, t_0+\tau] \mid dist_{1,2}(\mathbf{v}_{\mathbf{d}_{1|2}}, t) = c$,
- $det(\mathbf{v}_1, \mathbf{v}_{\mathbf{l}_{1|2}}) > 0$, $\|\mathbf{v}_1\| = \|\mathbf{v}_{\mathbf{l}_{1|2}}\|$, and $\exists t \in [t_0, t_0+\tau] \mid dist_{1,2}(\mathbf{v}_{\mathbf{l}_{1|2}}, t) = c$,
- $det(\mathbf{v}_1, \mathbf{v}_{\mathbf{r}_{1|2}}) < 0$, $\|\mathbf{v}_1\| = \|\mathbf{v}_{\mathbf{r}_{1|2}}\|$, and $\exists t \in [t_0, t_0+\tau] \mid dist_{1,2}(\mathbf{v}_{\mathbf{r}_{1|2}}, t) = c$.

The CIW $\mathbf{a}_{1|2}$, $\mathbf{d}_{1|2}$, $\mathbf{l}_{1|2}$ and $\mathbf{r}_{1|2}$ are then:

- $\mathbf{a}_{1|2} = (\mathbf{p}_1 + \mathbf{v}_{\mathbf{a}_{1|2}}.time(\mathbf{v}_{\mathbf{a}_{1|2}}),\ time(\mathbf{v}_{\mathbf{a}_{1|2}}))$,
- $\mathbf{d}_{1|2} = (\mathbf{p}_1 + \mathbf{v}_{\mathbf{d}_{1|2}}.time(\mathbf{v}_{\mathbf{d}_{1|2}}),\ time(\mathbf{v}_{\mathbf{d}_{1|2}}))$,
- $\mathbf{l}_{1|2} = (\mathbf{p}_1 + \mathbf{v}_{\mathbf{l}_{1|2}}.time(\mathbf{v}_{\mathbf{l}_{1|2}}),\ time(\mathbf{v}_{\mathbf{l}_{1|2}}))$,
- $\mathbf{r}_{1|2} = (\mathbf{p}_1 + \mathbf{v}_{\mathbf{r}_{1|2}}.time(\mathbf{v}_{\mathbf{r}_{1|2}}),\ time(\mathbf{v}_{\mathbf{r}_{1|2}}))$,

where $time(\mathbf{v})$ is the time of closest approach ($ttca$) delayed by a small amount: $time(\mathbf{v}) = ttca(\mathbf{v}) + min\left(\frac{c}{\|\mathbf{v}\|}, \frac{c}{\|\mathbf{v}_2\|}\right)$; $ttca(\mathbf{v})$ is computed assuming agents α_1, α_2 travel with velocities $\mathbf{v}$, $\mathbf{v}_2$.

This delay allows the two agents to get away from each other to avoid resetting them on a collision course if one of the two agents performs a new manœuver too soon (e.g., to solve one next interaction). The example presented in Figure 4 clearly shows that once the CIW is reached, the collision risk with the other agent is avoided.

3.4. collision-free Interaction Path

The collision-free Interaction Path construction is presented in Algorithm 1 and illustrated with an example in Figure 5. The collision-free IP is built iteratively by the algorithm. It starts with the straightforward IP to the goal (line 3). A collision test is performed with this IP (line 7). If no collision is found, a collision-free IP is detected (line 18). If a collision is found, the IP is discarded and the four CIW are computed and used to build four new IP (line 8). The test is then repeated on these new IP, first on the IS right before the interaction (line 9 to 12) then the IS from the interaction to the goal (line 16) until one IP reaches the goal (line 18).

```
   Data: Starting position s, Goal g
   Result: Build a collision-free IP
 1 checkList←[];
 2 collisionFreeList←[];
 3 add(sg, checkList);
 4 pathFound←FALSE;
 5 while !pathFound do
 6 |   bc←pullSegment(checkList);
 7 |   if collision(bc) then
 8 |   |   a,d,l,r ← computeCIW(bc);
 9 |   |   add(ba,checkList);
10 |   |   add(bd,checkList);
11 |   |   add(bl,checkList);
12 |   |   add(br,checkList);
13 |   else
14 |   |   add(bc, collisionFreeList);
15 |   |   if c!=g then
16 |   |   |   add(cg, checkList);
17 |   |   else
18 |   |   |   pathFound←TRUE
19 |   |   end
20 |   end
21 end
22 P←buildPath(s,g,collisionFreeList);
23 return P;
```

Algorithm 1: Algorithm used by EACS to build collision-free Interaction Paths

An example is presented in Figure 5. The initial situation is shown in Figure 5A): the blue agent starts at the position **s** and has to reach the position **g**. As we said the algorithm starts by testing the straight forward IP (Figure 5B)), a collision is detected and four new IP are created $\mathbf{sl}_{1|3}\mathbf{g}$, $\mathbf{sd}_{1|3}\mathbf{g}$, $\mathbf{sa}_{1|3}\mathbf{g}$ and $\mathbf{sr}_{1|3}\mathbf{g}$. Now, these four new IP have to be tested

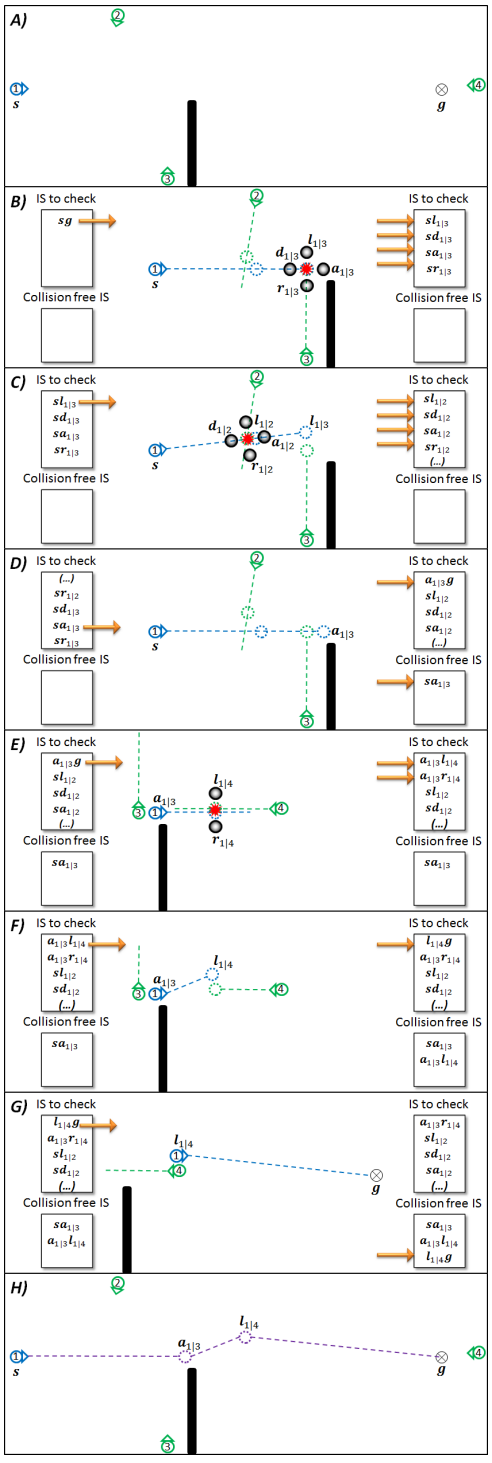

Figure 5: *Example showing the construction of a collision-free IP illustrating the different steps taken by the EACS system.*

starting with the IS just before the interaction: $\mathbf{sl}_{1|3}$, $\mathbf{sd}_{1|3}$, $\mathbf{sa}_{1|3}$ and $\mathbf{sr}_{1|3}$. Figure 5C) shows the test for the turning left solution $\mathbf{sl}_{1|3}$. A collision is detected on the IS and again four new paths are created. Figure 5D) shows the test for the accelerating solution $\mathbf{sa}_{1|3}$ and no collision is found. But this is only the first part of the IP, there is still the IS $\mathbf{a}_{1|3}\mathbf{g}$ to test. This test is done in Figure 5E). A collision is detected and only two paths are created. Both agents are going toward each other and cannot avoid the collision by accelerating. Decelerating will not solve the problem either, but only delay it as the blue agent will have to go toward the goal eventually. The turning left solution is tested on both IS, first on $\mathbf{a}_{1|3}\mathbf{l}_{1|4}$ (Figure 5F)) then on $\mathbf{l}_{1|4}\mathbf{g}$ (Figure 5G)), and no collision is found. The goal is reached and a collision-free IP is built: $\mathbf{sa}_{1|3}\mathbf{l}_{1|4}\mathbf{g}$ (Figure 5H)).

3.5. Cost and ranking

On the example presented in Figure 5, a collision-free IP $\mathbf{sa}_{1|3}\mathbf{l}_{1|4}\mathbf{g}$ is built. But if the turning right solution for the last interaction had been tested before the turning left one, we would have ended with the collision-free IP $\mathbf{sa}_{1|3}\mathbf{r}_{1|4}\mathbf{g}$. By testing the IS in a different order, we could have also built the collision-free IP $\mathbf{sl}_{1|2}\mathbf{g}$. In the end, there are many different IP to go through a crowd. But not all of them would be realistic, so we are looking for one that is efficient and would most likely be picked by a real walker. To that end, we used a cost function to rank the different IP and select a single one among them. We chose to use the energy consumption as the cost function. Real humans often favor the least energy consuming way of performing a task [Zip49]. Moreover it has already been used to improve collision avoidance models [GCC*10].

The energy formula, $E = m\int(e_s + e_w|\mathbf{v}|^2)\mathrm{d}t$, is from [Whi03]. The values of $e_s = 2.23J.Kg^{-1}.s^{-1}$ and $e_w = 1.26J.s.Kg^{-1}.m^{-2}$ minimize the energy consumption per unit of distance for a speed of $|\mathbf{v}| = 1.33m.s^{-1}$ which will become the preferred speed of the agent. We want to have agents with different preferred speeds. From the energy formula per distance, Equation (2), we set a constant distance to look for the speed that minimizes the energy consumption: $\mathbf{v}_{pref}$ (see Equation (3)). Then we are able to change the value of e_w for the chosen $\mathbf{v}_{pref}$ (see Equation (4)).

$$E = m\left(\frac{e_s}{|\mathbf{v}|} + e_w|\mathbf{v}|\right)d \tag{2}$$

$$\frac{dE}{dv} = m\left(\frac{e_s}{|\mathbf{v}|^2} + e_w\right)d = 0 \tag{3}$$

$$e_w = \frac{e_s}{|\mathbf{v}_{pref}|^2} \tag{4}$$

In some situations, allowing collisions is necessary to find good solutions (see Figure 12). So we enabled the search algorithm to consider IP with collisions. To keep a preference for collision-free IP, an extra cost is added for IP with collisions. This extra cost is the equivalent of making a long detour: the energy consumption at preferred speed for a specific distance which is a parameter of the system ($CollisionCost = m(\frac{e_s}{|\mathbf{v}_{pref}|} + e_w|\mathbf{v}_{pref}|)d_{collisionDetour}$). This way, agents will still go around dense areas when these are not too big, and will go through them otherwise.

3.6. Perception

When building an IP, the EACS considers every other agent for each step of the IP. This may lead to a combinatorial explosion. To reduce this risk and improve performance, we limit the number of agents considered by the EACS. This limitation is shared by real humans. For example humans have no eyes on their back and thus cannot perceive most of the people behind them. While their perception is limited, they are still capable of perceiving many people. We can assume that people consider only a fraction of them, that we call neighbors, when performing collision avoidance. In this section we describe a selection process to imitate human limitation and neighbor selection.

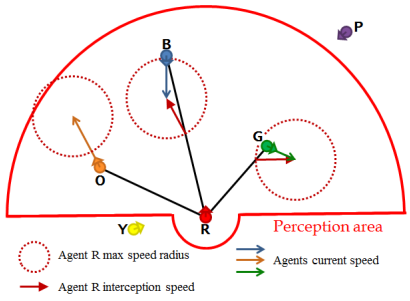

Figure 6: *Agent R perception system: First it selects the agents inside the perception area (O, B and G) then for each selected agent it computes the interception speed and selects the n agents with the smallest Minimum Interception Time (MIT) as neighbors (First B then G, O is not selected as it is going away too fast to be intercepted).*

Our selection process is inspired by previous ones (such as the one use in [vdBLM08]) and composed of two steps. The first step represents the physical limitation of human perception and consists of selecting all the agents in an area around the one performing collision avoidance. This area is represented in Figure 6 and is composed of two half circles: a big one in front of the agent and a small one behind. In the example from the Figure, Y and P are not selected as neighbors as they are outside of the perception area. The second step selects a limited number of perceived agents using a specific criterion. In our case we define the Minimum Interception Time (MIT) in Equation 6 and select agents with the lowest

MIT value. Indeed, the harder it is to intercept another agent, the less likely it is to cause a collision with any IP. This allows us to select fewer agents than with the regular euclidean distance criterion while conserving good behaviors.

$$IT_{1|2} = \{t \in \mathbb{R}^+ \mid \exists \mathbf{v} \in \mathbb{R}^2 : \|\mathbf{v}\| \leq MS_1, \mathbf{p}_1 + \mathbf{v}t = \mathbf{p}_2 + \mathbf{v}_2 t\} \quad (5)$$

$$MIT_{1|2} = min(IT_{1|2}) \quad (6)$$

In Figure 6, B would be selected before G despite G being closest. As for O, it would not be selected as it is going away too fast to be intercepted.

3.7. Following case

When there is someone in front going in the same direction but slower, there is not always enough space to avoid them. In this case, people can try to squeeze to avoid them despite the lack of space or they can match the speed of the person in front to follow them until there is enough space to overtake them or until their trajectories separate. The following behavior is close to collision avoidance as it can be used to prevent a collision. At the same time, it is very different as the collision is never solved by following, only postponed. As said before, people follow until they can properly avoid the person in front of them or the collision risk disappears by itself. When computing the four CIW, the agent follows a velocity that prevents the collision until the collision risk no longer exists. As explained above, when following, the collision risk remains except if one of the agent changes direction. So the agent can still change its speed to match the one of the agent in front and prevent a collision by following, but we need to determine how long it should follow to compute the corresponding IW.

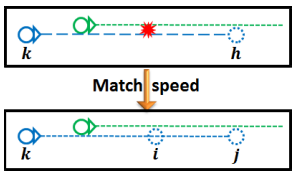

Figure 7: *Detection of a following case and the two Interaction Waypoints* **i** *and* **j** *computed from the following speed.*

Following happens when the tested IS has a collision with an agent that goes in the same direction, this situation is represented in Figure 7. Whether the IW **h** (the end of the IS) is the goal or just an intermediate IW placed here to avoid another agent, the position and/or the time of the IW is important. This is why it has been decided that when following, two IW should be computed. For the first one **j**, the agent will follow until it reaches the previous position he wanted to reach (the position of **h**). For the second one **i**, the agent will follow for as long as it would have walked with the previous IS **kh**.

From the Figure 7, we have the velocity of the green agent **v**, the IW $\mathbf{k} = (\mathbf{p_k}, t_\mathbf{k})$ and $\mathbf{h} = (\mathbf{p_h}, t_\mathbf{h})$. We then compute the IW $\mathbf{j} = (\mathbf{p_h}, \frac{\mathbf{p_h} - \mathbf{p_k}}{\mathbf{v}} + t_\mathbf{k})$ and $\mathbf{i} = (\mathbf{v} * (t_\mathbf{h} - t_\mathbf{k}), t_\mathbf{h})$.

3.8. Extended search

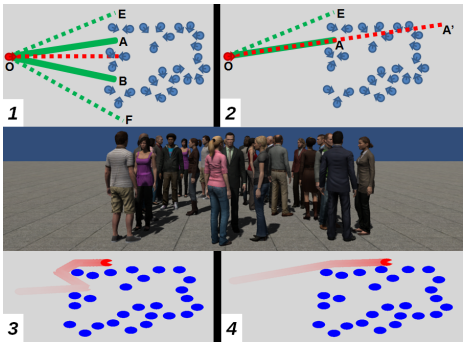

Figure 8: *1) Without extra search, the algorithm stops at the IS* **OA** *and never considers* **OE**. *2) With extra search, the IS* **OE** *is created by solving the collision on the IS* **OA**$'$. *3) Agent's trajectories without extra search, the agent starts to go inside the crowd before backing up and going around. 4) With extra search, the agent is able to directly go around the crowd.*

The construction of the IP is done step by step. When a collision is found, IS that solve the collision are searched in several directions (by turning or by adapting speed). Once such a segment is found, no other solution is searched in the current direction. If we consider the situation in Figure 8, the stop of the search can lead to unnatural behaviors. In this situation, we considered the IP built by the red agent avoiding all the blue agents. The blue agents are all standing in order to simplify the representation of the IS as we do not have to consider their movement through time and collision avoidance solution can only be found by turning. As said, once an IS is found solving a collision avoidance, the search is stopped in the current searched direction. In the current situation, it means that only the IS **OA** and **OB** in Figure 8.1 will be considered as first IS of the IP. In the end, the agent will be able to only consider a path that goes inside the crowd and exits it again which produces the results presented in Figure 8.3.

As we can see on the first person view, an obvious solutions would be to go around the entire crowd directly which means that the agent needs to also consider **OE** and **OF** as first IS. In order to extend the search to find these two IS, we added a few extra steps to the algorithm. When an IS free of collision is found, another IS is considered with the same speed for a longer period of time. If no collision is found, then the IS represents a way out of the crowd that does not require to go back on its steps, so the search returns to its normal process. If a collision is found, then we insert the

new IS with the collision to the search process to continue searching for more solution in the same direction. In the example in Figure 8.2, the IS **OA** is extended to the segment $\mathbf{OA'}$, then the collision on $\mathbf{OA'}$ is solved by turning left (as turning right would bring us back in the search pattern) and the IS **OE** is created.

As a result, the agent is now able to avoid going inside a crowd if it is only to backtrack on its own path. This fixed behavior is presented in Figure 8.4. This extra search has been implemented only for the first IS of the IP. This has the advantage of limiting the extra computation time required by the extra searched paths. This extra search can be extended to the others IS of the IP, but the first interaction is the most important one as it gives the current direction to the local avoidance system and is enough to prevent going inside a dead end situation if there is a better way around.

4. Results

In this section we present the results of our work. The simulations have been done by using our system coupled with RVO2. EACS computed a strategy and then set the preferred velocity required by RVO2. No additional pathfinding algorithms have been used as the studied situations were simple enough and the goal is always reachable from the agents' starting position.

4.1. Performance

The EACS system computes an IP and has to explore many possibilities to find a good one. This exploration leads to a combinatorial explosion as for each constructed IP, a collision detection is done with the others agents and when a collision is found four new IP are created which in turn will need collision detection and might create 4 more IP. Several methods have been used to improve the system's performance:

- **Smart exploration:** an estimation of the final cost of an IP is done to advance the construction of the most promising one in priority.
- **Limited exploration:** when a goal is far away, it is very unlikely that real humans will fully plan collision avoidance to the goal. Moreover, planning too far in the future becomes quickly ineffective as the simple prediction of constant velocity becomes erroneous. The EACS exploration is stopped when the IP reaches a number of consecutive interaction that is too big.
- **Late update:** much like path planning, the interaction planning does not need to be done every step. While the environment in the interaction planning is not static and thus IP need to be regularly updated, IP can be reused for several steps. This allows us to spread the computational cost of all the agents' planning on several step similarly to what has been done in [SKH*11].
- **Neighbor selection:** the perception system presented in section 3.6 reduces the number of agents considered when computing collision detection which greatly improves computation time.

The two methods with the best impact on computation time are the Late update and the Neighbor selection. On a $20s$ simulation of 400 agents in a four flow crossing, the simulation took $60.3s$ to compute without any neighbor selection. With a simple neighbor selection based on the area only, the computation time dropped to $35.4s$. When limiting the number of neighbors to 100, the computational time became $3.74s$ making it possible to simulate larger numbers of agents. On a $20s$ simulation of 1000 agents in a four flow crossing, the simulation took $35.7s$ to compute when updating the plan at every step for each agent. When updating the interaction plan every 10 steps, the computational time dropped to $5.19s$.

Performance tests have been run with 4 different situations:

- **One-Way Flow (OWF):** one flow with a starting density of $0.9 agents.m^{-2}$ moving in one direction with agents having different preferred speeds. The size of the flow is variable (bigger flow, same initial density)
- **Bidirectional Flows (BF):** two symmetrical flows with a starting density of $0.9 agents.m^{-2}$ moving in opposite direction with agents having different preferred speeds. The size of the flows is variable (bigger flow, same initial density)
- **Four Crossing Flows (FCF):** four flows with a starting density of $0.9 agents.m^{-2}$ crossing each other at the same position with agents having different preferred speeds. The size of the flows is variable (bigger flow, same initial density)
- **Sandbox (SB):** 1000 agents randomly moving in a square, when an agent is close to its current goal a new one is randomly chosen on the square border. The size of the square is variable to simulate different densities.

All the tests have been run on a computer with an Intel®Xeon®Processor E5-1603 (4 cores, 10M Cache, 2.80 GHz, 0.0 GT/s Intel®QPI) and 32GB Memory with the following set of parameters: updateFrequence=10, MaxInteractions=9. Results are presented in figure 9.

As can be seen on the graphs in Figure 9, adding the EACS to RVO2 significantly increase the computation time. Nevertheless, thanks to the different mechanisms that reduce the complexity, EACS is able to simulate several thousands of agents in real time (computation time$< 20s$ for 5000 agents). It is true even for complex situations (FCF) as complexity does not influence computational time with large numbers of agents (≈ 5000).

While the size of the simulation is the main factor influencing computation time, density also has a significant impact. This impact is visible on the right graph in Figure 9. It

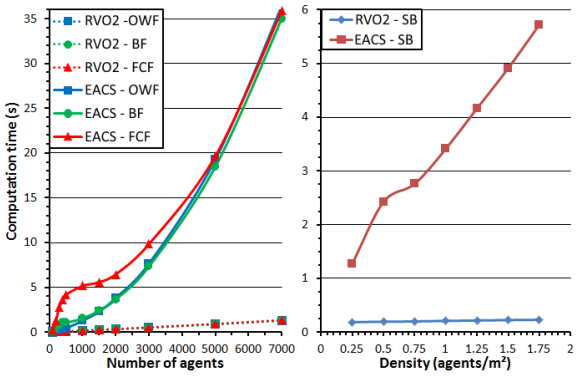

Figure 9: *The graphs show the time needed to compute 20 seconds of simulation for the different performance test situations with and without EACS. On the left graph, the results of the three situations: OWF, BF and FCF are presented for different numbers of agents. The right graph presents the results of the SB situation for different densities.*

seems that computation time increases linearly with density. Indeed, it is harder to plan ahead in high density. Moreover, collisions appear more often causing IP to be smaller. Plans are update every 10 steps except if the agent reaches the end of the current IP. This happens more often with smaller IP.

4.2. Case study

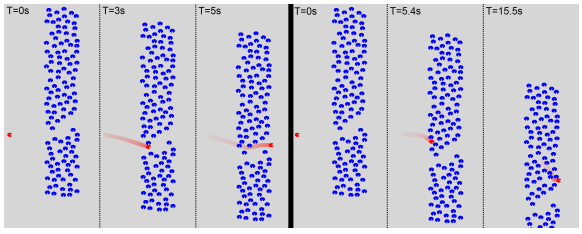

Figure 10: *Simulation of an agent (in red) going through a flow, with the flow having a tunnel to facilitate the crossing. Left: simulation including EACS; Right: simulation without EACS .*

Several situations where designed to test the EACS system and check the improvement it brings to crowd simulation. The first one is the example used in the introduction to show the limitation of the current models. The results are shown in Figure 10. As expected, RVO2 alone is unable to use the tunnel to go through the flow. Instead, the red agent goes straight forward and has trouble going through the flow as it spends around 10 seconds in the flow itself. With the EACS system, the red agent is able to take the tunnel which facilitates its navigation greatly: the agent spends only 2 seconds in the flow.

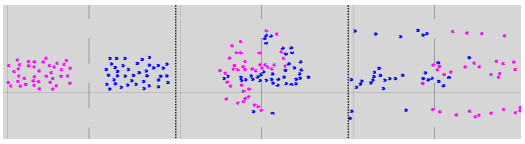

Figure 11: *Simulation of a clogged exit, EACS is able to choose other exits that are further away but not clogged.*

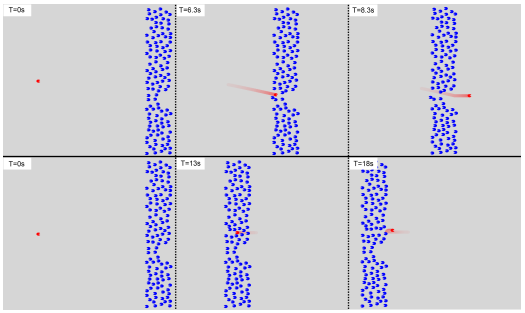

Figure 12: *Simulation of an agent (in red) going through a flow that goes in the opposite direction. The flow has no tunnel but a section of it has a lower density. At the top, we have the simulation with EACS and at the bottom the simulation without EACS.*

The second one is shown in Figure 11 and is very similar to the first one. Instead of going through a flow, agents need to exit or enter a building that has 3 entrances. All the agents are close to one exit which quickly becomes clogged. As for the first situation where the agent was able to detect the tunnel, in this situation, some of the agents detect that the other two exits are empty. Instead of waiting and avoiding many obstacles which can be painful, they decide to go through the two others exits. The extra cost from the detour is largely compensated by all the required velocity adaptations to go through the clogged exit. This is easily witnessed in real life: in the train station for example, where people will use further exits to avoid the dense crowd of the closest one.

In the third situation shown in Figure 12 there is no collision-free solution to cross the incoming flow. Such situations exist in real life where we have to cross a very dense crowd and we require other people to give us some space to go through. In the presented situation, even if there are no collision-free paths through the crowd, there is a weak point where the density is lower. Going through this point would make it easier to go through the flow. EACS is able to detect such weak points in the flow and compute a path through it. If we compare with the trajectory computed without EACS (at the bottom), we can clearly see that it is easier to use the weak point to go through the flow. In the end, the agent controlled by RVO2 spends around 10 seconds in the flow while the one controlled by EACS spends only 2 seconds in it.

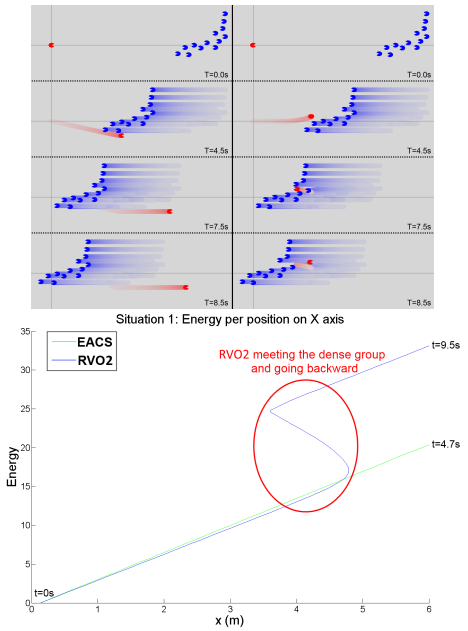

Figure 13: *Top: simulated test where short-term cost leads to bad mid-term cost, agents simulated with EACS on the left and without on the right. At the bottom, the graph quantifies the energy consumption of the red agent (For RVO2 and EACS) as they walk towards the goal (In this situation, it is equivalent to the distance traveled on the x axis). The section of the blue curve circle in red corresponds to RVO2 agents going backwards to avoid the dense group (T=7.5s on the top figure).*

Another situation is shown in Figure 13 and has been designed to show the difference in optimizing the short-term outcome and the mid-term one. If we look at the trajectory given by RVO2 alone (top right), we can see that the red agent is avoiding by going on the left. If we consider only the first few blue agents, it is the best solution as going on the right would require to go faster and to shift more on the side. This is confirmed by the energy graph, the energy consumption of EACS which goes on the right is higher than the one of RVO2 for the first $3m$. But when we look further, we can see that the agent will meet with more people when going to the left. This dense area is so hard to navigate that the agent will even move backwards. Looking at the graph, we can notice the sudden rise in energy consumption for RVO2 around $4m$ as well as the curve going backwards because the agent walks in the opposite direction of its goal. The curve of EACS energy consumption remains stable which confirms that the EACS strategy is better on the mid-term.

The last situation is about group avoidance. Previous experiments have shown that real humans go around dense groups and through sparse ones [BOP15]. We have made several simulations where an agent had to avoid groups of

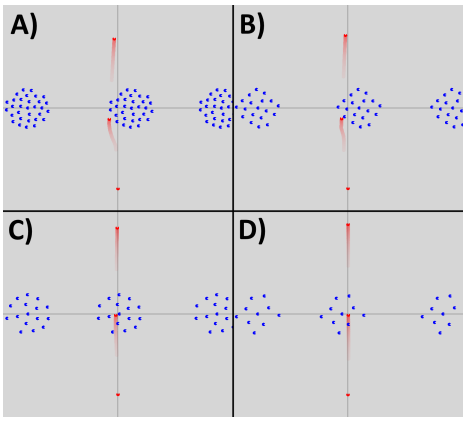

Figure 14: *Simulation of group avoidance for different group densities.*

different densities to check that EACS is able to reproduce the real trend. We can see some of them in Figure 14, the situation setup is the same as the one in the experiment with real humans. The red agent is able to decide whether to go through the group or around it and reproduces the same trend as that of a real human: it circumvents the two dense groups (Figures 14A and 14B) and traverses the sparse ones (Figures 14C and 14D).

4.3. Energy

To check the energy gain of EACS over pure RVO2, we studied the trajectories of agents going through an orthogonal flow. We tried several different conditions to analyze their impact on the results. We performed 2800 simulations: 7 density types x 19 exploration limits for EACS and 1 without EACS x 20 repetitions. For each simulation, a flow was randomly generated and 10 agents were randomly placed on the same side of the flow with their goal on the other side. The energy consumption of the 10 agents was computed from their starting position to their goal location and then averaged.

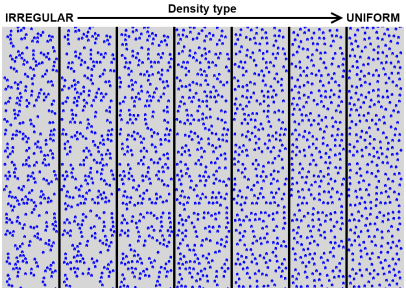

Figure 15: *The different density types used for the energy study from very irregular (left) to uniform (right).*

The first condition studied is the density type. For each generated flow, the same dimensions (10mx80m) and number of agents (700) were used. Thus the global density was the same for each flow. The variation among the different conditions was made on the regularity of the density in the flow. These variations are shown in Figure 15. We have a uniform flow on the right and the different types of irregularity until the very irregular flow on the left. These irregularities create small dense clusters and leave some space in between to navigate more easily. We expect EACS to be able to use this in-between space and perform better when going through irregular flows.

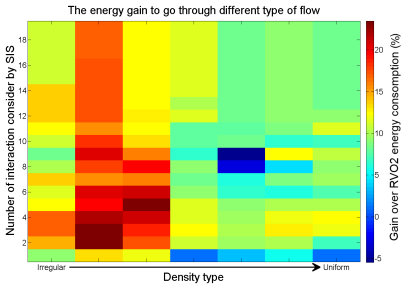

Figure 16: *This graph shows the amount of energy preserved by the agents when using EACS to optimize their avoidance strategies compared to EACS-less strategies.*

The second condition studied is the exploration limit for EACS. When EACS is looking for a path it will sequence several interactions over time. The number of interactions can become very big in some situations which can cause performance issues. Moreover, it is doubtful that real humans can consider a high number of future interactions. For this reason, we set a maximum number of interactions when constructing an IP in EACS. When an IP reaches the limit it will not be extended anymore. We expect that a low maximum number of interactions will lead to results close to RVO2. We also expect that a very high number of interactions will perform similarly or worse than a medium number of interactions: the further we plan in the future, the less the extrapolation is relevant.

For the results, for each exploration limit, we compare the average energy consumption with and without EACS. The Figure 16 shows this consumption difference in percent. As expected, EACS is performing better for very irregular densities. When the density becomes uniform, the energy consumption with and without EACS becomes similar. If we look at the different results for different maximum numbers of interactions, we can see that having a value higher than 14 does not seem to change the results much. Indeed, optimum results seem to be found for a maximum number of interactions around 4.

4.4. Limitations

Our system has limitations. The main one is the number of future interactions explored by the system. We showed in the previous section that the system is particularly efficient when used in specific contexts, such as when agents' density is not uniformly distributed in space. We can actually easily guess what type of crowd motions will enable EACS to be efficient. For example, we expected that non-uniform density distributions would open larger efficient interaction paths in the crowd with a great probability that these paths remain valid in spite of all agent adaptations.

More generally, the EACS system is particularly efficient when the initial predictions are good and turn out to be representative of the actual motion of neighbor agents. In the opposite case, predictions turn out to be false, and the energy gain by EACS is poor. We also observe in the energy consumption evaluation of EACS solutions (previous section) that considering interactions too far in the future decreases the quality of the solutions provided by EACS. Note that EACS guided agents toward fewer counterproductive situations than naive short-term solutions.

One solution to this could be to evaluate on-line the quality of the predictions made by EACS. The number of interactions explored in the future could be directly dependent on the quality of this prediction. If predictions are good, agents can explore wider future time windows and more numerous future interactions, and conversely.

Another limitation of EACS is to only consider an energy-related cost function. We are convinced that this criterion is considered by human walkers when setting mid-term strategies, but we are also convinced that other factors play a role. For example, we could check how close the solution paths are to other agents' motion. Additional social distances could be considered in the cost of paths. As another example of a social criterion, passing in front of an agent at a close distance could be penalized in comparison with the strategy of giving way, which could be considered to be more polite.

Finally, EACS explores a subset of all the possible strategies to perform collision avoidance. We make an arbitrary choice to explore adaptations exclusively made of speed or orientations changes, whereas all intermediary solutions could be explored. We are here in a typical trade-off between computation times and quality: we could add more than 4 CIW in the system to consider mixed-adaptation strategies, but this would result in higher computation times.

5. Conclusion

We have modified the usual navigation process of crowd simulation to add some mid-term considerations. To this end we have designed an interaction planner, the Effective Avoidance Combination Strategy that creates interaction pathes which enable agents to consider non-straight paths to

go through a crowd. As a result, we have shown with specific test cases that the agents are able to select strategies that use density tunnels to facilitate their navigation through a crowd. Moreover, while building the interaction path, EACS evaluates the repercussion of its avoidance choices further in the future. This allows agents to make some extra effort in resolving their current collision so as to simplify future interactions.

With an energy study, we show that by adding EACS to the regular navigation process, agents use less energy to navigate. The study has shown that EACS is especially useful to improve trajectories when facing crowds with irregular densities. The study has also highlighted that too much midterm planning is not beneficial, which can be explained by the incapacity to correctly extrapolate the other agents' trajectories too far in the future. It seems that planning around 4 interactions ahead yields the best results.

We have shown the improvement in the quality of agents' navigation brought by our work. The next step is to compare it with real human behaviors and interaction planning. Especially, it would be interesting to study how many interactions ahead real humans plan. Another interesting direction is to work on the evolution of the interaction path. Right now, the IP is computed at a constant interval and each time the computation starts from scratch. Real humans are known to be highly adaptable creatures, in most cases they most likely adapt their plan based on changes in the environment instead of planning a new one from scratch. This would greatly improve performance, but it might also improve results by making the plan more reactive to the environment changes.

Acknowledgements

This work is funded by the French National Research Agency ANR, project PERCOLATION number ANR-13-JS02-0008.

References

[BLA02] BAYAZIT O. B., LIEN J.-M., AMATO N. M.: Roadmap-based flocking for complex environments. In *Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on* (2002), IEEE, pp. 104–113. 2

[BNCM14] BEST A., NARANG S., CURTIS S., MANOCHA D.: Densesense: Interactive crowd simulation using density-dependent filters. In *Symposium on Computer Animation* (2014), pp. 97–102. 3

[BOP15] BRUNEAU J., OLIVIER A.-H., PETTRÉ J.: Going through, going around: a study on individual avoidance of groups. *Visualization and Computer Graphics, IEEE Transactions on PP*, 99 (2015), 1–1. 10

[FS98] FIORINI P., SHILLERT Z.: Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research 17* (1998), 760–772. 4

[GCC*10] GUY S. J., CHHUGANI J., CURTIS S., DUBEY P., LIN M., MANOCHA D.: Pledestrians: A least-effort approach to crowd simulation. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2010), SCA '10, Eurographics Association, pp. 119–128. 3, 6

[GKGG14] GODOY J., KARAMOUZAS I., GUY S. J., GINI M.: Anytime navigation with progressive hindsight optimization. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Sept 2014), pp. 730–735. 3

[GNCL14] GOLAS A., NARAIN R., CURTIS S., LIN M.: Hybrid long-range collision avoidance for crowd simulation. *Visualization and Computer Graphics, IEEE Transactions on 20*, 7 (July 2014), 1022–1034. 3

[HFV00] HELBING D., FARKAS I., VICSEK T.: Simulating dynamical features of escape panic. *Nature 407*, 6803 (2000), 487–490. 2

[KBG*13] KAPADIA M., BEACCO A., GARCIA F., REDDY V., PELECHANO N., BADLER N. I.: Multi-domain real-time planning in dynamic environments. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2013), SCA '13, ACM, pp. 115–124. 2, 3

[KGO09] KARAMOUZAS I., GERAERTS R., OVERMARS M.: Indicative routes for path planning and crowd simulation. In *Proceedings of the 4th International Conference on Foundations of Digital Games* (2009), ACM, pp. 113–120. 2

[KSHF09] KAPADIA M., SINGH S., HEWLETT W., FALOUTSOS P.: Egocentric affordance fields in pedestrian steering. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2009), I3D '09, ACM, pp. 215–223. 3

[PCM*06] PETTRÉ J., CIECHOMSKI P. D. 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 (2006), 445–455. 2

[Rey87] REYNOLDS C. W.: Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph. 21*, 4 (Aug. 1987), 25–34. 2

[SGA*07] SUD A., GAYLE R., ANDERSEN E., GUY S., LIN M., MANOCHA D.: Real-time navigation of independent agents using adaptive roadmaps. In *Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2007), VRST '07, ACM, pp. 99–106. 2

[SKH*11] SINGH S., KAPADIA M., HEWLETT B., REINMAN G., FALOUTSOS P.: A modular framework for adaptive agent-based steering. In *Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2011), I3D '11, ACM, pp. 141–150 PAGE@9. 8

[ST05] SHAO W., TERZOPOULOS D.: Autonomous pedestrians. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), ACM, pp. 19–28. 2

[vdBLM08] VAN DEN BERG J., LIN M., MANOCHA D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* (May 2008), pp. 1928–1935. 2, 6

[Whi03] WHITTLE M. W.: *Gait analysis: an introduction*. Elsevier, 2003. 6

[Zip49] ZIPF G. K.: *Human Behaviour and the Principle of Least-Effort*. Addison-Wesley, Cambridge MA, 1949. 6