EUROGRAPHICS 2016 / J. Jorge and M. Lin
(Guest Editors)

Volume 35 (2016), Number 2

Gradient-based steering for vision-based crowd simulation algorithms

T. B. Dutra[1†], R. Marques[2,3†], J. B. Cavalcante-Neto[1], C. A. Vidal[1] and J. Pettré[2]

[1]Universidade Federal do Ceará, Brazil

[2]INRIA Rennes - Bretagne Atlantique, France

[3]Universitat Pompeu Fabra, Spain

Abstract

Most recent crowd simulation algorithms equip agents with a synthetic vision component for steering. They offer promising perspectives through a more realistic simulation of the way humans navigate according to their perception of the surrounding environment. In this paper, we propose a new perception/motion loop to steering agents along collision free trajectories that significantly improves the quality of vision-based crowd simulators. In contrast with solutions where agents avoid collisions in a purely reactive (binary) way, we suggest exploring the full range of possible adaptations and retaining the locally optimal one. To this end, we introduce a cost function, based on perceptual variables, which estimates an agent's situation considering both the risks of future collision and a desired destination. We then compute the partial derivatives of that function with respect to all possible motion adaptations. The agent then adapts its motion by following the gradient. This paper has thus two main contributions: the definition of a general purpose control scheme for steering synthetic vision-based agents; and the proposition of cost functions for evaluating the perceived danger of the current situation. We demonstrate improvements in several cases.

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

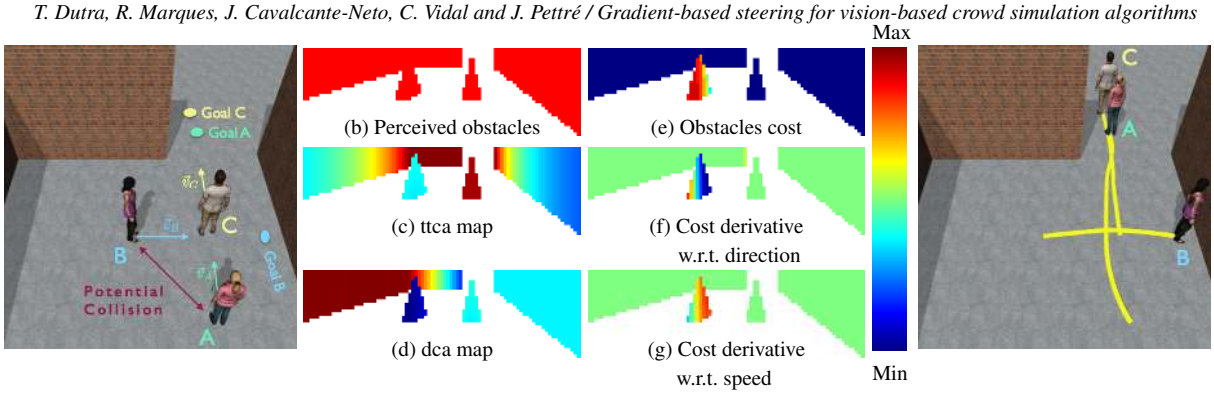
1. Introduction

Crowd simulation is rapidly extending over various application fields (e.g., entertainment industry, architectural design, etc.) which demand a high level of visual quality and/or realism. Our paper focuses on microscopic approaches which formulate the crowd simulation problem at the individual scale. The core of these approaches is the model of local interactions which defines how each agent's motion is influenced by other neighbor agents and obstacles. The search for the best possible model is thus of prime importance to improve the quality of crowd simulators. However, due to performance constraints, only a small amount of computing time can be devoted to the task of steering an agent. Consequently, the main difficulty is to keep simple model formulations, while generating artifact-free motions.

The crowd simulation field has been very active in the recent years and proposed many models of local interactions, most of them dedicated to collision avoidance. They can be divided into two categories: position-based and velocity-based approaches. The former maintain agents' clearance by controlling them based on a distance field (either explicitly computed [TCP06] or not [HM95]). The latter steer agents so that their velocity does not lead to a collision in the near future. Both approaches generate artifacts of different natures. A well known artifact for position-based methods is the lack of anticipation leading to non-natural individual trajectories in mid-density conditions or instabilities in higher ones. A typical artifact of velocity-based approaches is produced by the partition of the agents' velocity space into admissible and non admissible velocities, depending on whether or not they lead to collisions in the near future. As a consequence, agents are allowed to move at quasi-contact distances which can cause non-natural spatial distributions of agents during motion.

The key idea of this paper is to revisit the motion control scheme for velocity-based algorithms. Instead of adapting trajectories if and only if a collision is predicted, we steer agents by constantly minimizing the risk of collision. We consider a continuous definition of the risk of collision so that increasing the future distance of closest approach beyond the contact distance will decrease the risk of future collision. We build a solution based on a synthetic vision technique. Our objective is to make the motion control loop more

† Teófilo Dutra and Ricardo Marques are joint first authors



(a) Initial state (h) Final state

Figure 1: *The mechanics of steering agents with our gradient-based synthetic vision model. (a) Three agents walk toward their goal, with a potential collision between agents A and B. (b) obstacles detected by agent A for the situation depicted in (a). (c) and (d) show the time to closest approach (ttca) and distance of closest approach (dca) for each perceived obstacle (blue encodes a low value, while red corresponds to a high value). Obstacles with low ttca and dca convey a significant risk of collision, which leads to high cost in (e). To solve the problem, the agent determines the partial derivatives of the obstacles cost w.r.t. direction and speed ((f) and (g), respectively). Collision is avoided by descending the gradient so as to reduce the cost. The resulting trajectories are shown in (h).*

robust to complex situations of collision avoidance by adopting a locally optimal steering approach. To this end, we define a cost function that is used to characterize the current situation according to the visual information, accounting for both the danger of all visible obstacles and the agent's goal. At each time-step, the cost function and its gradient are evaluated. Agents adapt their motion by resorting to a gradient, in order to minimize the cost function, which is equivalent to selecting a motion that corresponds to the best trade-off between reaching the agent's goal and reducing the perceived danger of the situation.

The contribution of this paper is twofold. The first is the definition of a cost function, used to evaluate both the risk of future collision of the agent, as well as whether or not it is heading towards its goal. Instead of a binary definition of collision risk, as used in previous velocity-based approaches, we consider the danger of passing close to obstacles, based on the notion of closest distance of future approach. Our second contribution is a control method that steers agents resorting to a gradient descent of this cost function. As opposed to previous position-based methods, which mapped position changes (velocity) to the gradient of a distance field, we map accelerations to the derivatives of our cost function. In addition, our algebraic formulation avoids the need for explicit optimization of the cost function. The remainder of this paper is organized as follows. In Section 2, we discuss the most relevant related work. In Section 3, we present an overview of the model, the details of which are given in sections 4, 5 and 6. In Section 7, we describe the most important aspects of the implementation and we report the influence of the parameters on the model. The results are presented in Section 8, which is followed by a discussion and some conclusions.

2. Related work

The main objective of crowd simulation is to compute the motion of many characters which results from collective behaviors. This objective has received a wide attention from various disciplines and many solutions can be found in the literature [PAB08, TM13, ANMS13]. These solutions can be classified in two distinct groups: macroscopic and microscopic approaches. Macroscopic approaches consider a crowd as a continuous mass and simulate its motion based on conservation laws. Models based on guidance fields [Che04, TCP06, NGCL09, PvdBC*11] are representative examples. The microscopic approaches, on the other hand, model local interactions between agents which influence each other's motion. Collective behaviors and global patterns emerge as a consequence of the agent's local interactions. Our new model falls in the category of microscopic crowd simulation algorithms.

A large number of microscopic algorithms have been proposed. Example-based approaches create a crowd motion by concatenating or editing pre-recorded trajectories [LCL07, KLLT08] or by learning the relations between motion variables during specific types of interactions [JCP*10]. Those approaches preserve the realism of local trajectories, but do not generalize as well as the microscopic approaches which try to mathematically formulate the main principles by which humans move in crowds. This category of approaches provides an interesting trade-off between animation control, computational efficiency and animation quality. In this latter category, some approaches formulate a set of rules [Rey87, ST07, SKH*11], whereas other authors resort to forces such as those of a particle system [HM95, HFV00]. Recently, velocity-based approaches have received increased attention. Such models anticipate the danger by extrapolating the agents' trajectories to detect potential collisions in a near future. Extrapolating trajectories for collision prediction is not new. In [Rey99], Reynolds presented the unaligned collision avoidance behavior for this purpose and, in the following years, other works on this direction have been presented [PPD07, vdBLM08, KHvBO09, POO*09, KSHF09, GCK*09, KSH*12]. The most recent evolution of velocity-based obstacles is represented by the Optimal Reciprocal Collision Avoidance (ORCA) approach [vdBGLM11]. It efficiently computes the optimal solution in the velocity-space by extrapolating trajectories to predict potential collisions, hence re-

ciprocally avoiding collisions between agents in a near future. A particularly interesting velocity-based approach is the one of [OPOD10], based on Synthetic Vision and inspired by literature which acknowledges the role of the human vision system in the locomotion perception-action loop [CVB95, WF04, RRW14].

Despite the recent improvements, crowd simulation algorithms still suffer from important drawbacks. The reasons are many and depend on each particular approach. In this paper we explore two of them. First, the reactive process by which interactions are solved (e.g., [Rey99, OPOD10, KHvBO09]): agents react to a collision danger with a basic and unique response, which drastically limits the range of possible strategies. This unique solution is generally relevant in the simple case of a single interaction, but may turn into a totally inefficient strategy in the general case of multiple interactions combination. Furthermore, residual collisions are often observed since agents only focus on avoiding the most imminent danger and neglect the consequence of their motion adaptations on other potential dangers. Finally, although these strategies often yield emergent patterns under specific traffic conditions (as expected), those patterns often seem too strong and unnatural, probably because the agents tend to act similarly. Velocity-based approaches provide an answer to this problem because they combine interactions to determine the full set of admissible velocities that allow agents to avoid any collision with any agent in their surroundings in the near future. But this paradigm raises the second problem we explore in our work. The agent's motion will be steered with velocities guiding them at the limit of future contact with other agents. This does not fully capture human behaviors which may adapt the interaction distance depending on the context.

In this paper, we present a new model aimed at overcoming most of those limitations. Unlike existing approaches which make agents react only if a risk of contact is predicted, our agents adapt their motion even if no collision is detected so as to further diminish the risk of future collision. This new principle is implemented as a cost function used to continuously perform locally optimal adaptations. Previous gradient descent methods existed in the literature (e.g., [TCP06, HFV00]), but they all proposed cost functions which relate agent position to the distance to other agents. Most recent gradient based approaches also explored time-to-collision based functions [KSG14]. In contrast, our cost function relates risk of future collision with velocity adaptations. This way, like velocity-based techniques, we enable anticipated reactions to solve interactions. Moreover, we avoid the problem raised by the dichotomous partition of the reachable velocity space into admissible and colliding components, since our method evaluates the risk of collision through a continuous cost function, as explained in the next section. Our proposal also has some resemblance to the egocentric field approach [KSHF09, KSH*12] given that both approaches resort to a control loop for steering agents. In their approach, the control is composed of sensory, affordance and selection phases. This could be associated to our perception, evaluation and action phases (which will be detailed later). However, our approach differs in several aspects. The major difference is the way the risk of future collision is evaluated and minimized. The Egocentric field performs a space-time local motion planning to evaluate an optimal solution, whereas our approach evaluates a solution directly from instantaneous motion data.

3. Overview

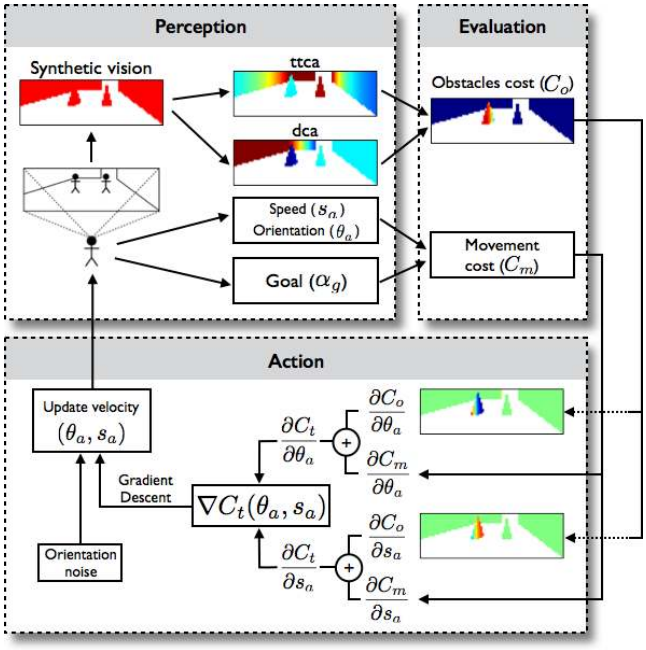


Figure 2: *The 3-phase control loop.* ***Perception:*** *the agent perceives the surrounding environment resorting to its synthetic vision.* ***Evaluation:*** *the agent assesses the cost of the current situation.* ***Action:*** *the agent takes an action to reduce the cost of the situation.*

This section gives an overview of the control loop according to which agents are steered in our simulation (Section 3.1), as well as the mathematical notation (Section 3.2).

3.1. Control loop

We propose a new control loop to steer agents in crowded environments with static and moving obstacles. An example of a simulation result is depicted in Fig. 1. Fig. 1 (a) shows the environment and the initial state of three agents that should move toward their goals. The motion of agents *A* and *B* presents a risk of collision. With our model, agent A changes its linear trajectory and adapts its velocity in order to pass behind agent B (Fig. 1 (h)). Agents are steered to reach their goals and to avoid collisions according to the loop shown in Fig. 2. The control loop is composed of three phases: *perception*, *evaluation* and *action*. The complete 3-phase loop is performed for each time-step of the simulation and for each agent.

In the *perception* phase each agent perceives the relative motion of neighbor obstacles (both static and dynamic). In our approach, we model each obstacle as a set of points and estimate the relative motion for each of these points (under the hypothesis of constant velocity for agent and obstacle, just as any other velocity-based approach). We then compute, for each point, the time to closest approach (*ttca*) and the distance of closest approach (*dca*). The perception phase is detailed in Section 4. Fig. 1 (b) illustrates the environment as perceived by agent A in Fig. 1 (a). Figs. 1 (c) and (d) show the respective *ttca* and *dca* maps.

The role of the *evaluation* phase is to estimate how 'good' the agent's current velocity is, given the risk of collision with the perceived obstacles and the alignment with the goal. Such evaluation is made through the definition of a cost function C_t, which consists of two components: the obstacles cost C_o, which captures the risk of collision based on the perceived $ttca$ and dca values; and the movement cost C_m, which considers the task of reaching a goal based on the agent's speed s_a, orientation θ_a and angle with the goal α_g. The evaluation phase is detailed in Section 5. Fig. 1 (e) illustrates the C_o associated with each pixel. Note the high risk of collision with agent B given by the high obstacle cost values.

The role of the *action* phase is to update the agent's velocity to minimize the cost function C_t. To this end, the partial derivatives of C_t with respect to the agent's motion variables are computed and the locally optimal move is deduced. The action phase is detailed in Section 6. Fig. 1 (f) and (g) illustrate the partial derivatives of C_o for each pixel.

The agent's change of orientation and speed is determined through the collective information of the gradient of the cost functions computed at each pixel. Thus, the values illustrated in Fig. 1 (f) and Fig. 1 (g) induced agent A to perform a left turn and to reduce its speed in order to avoid collision with agent B. A small noise ε is added to the agent's new direction θ_a so as to disrupt symmetry as shown in Fig. 2.

3.2. Mathematical characterization of the agent's state

Table 1: *Description of symbols. Bold face notation represents vectors, otherwise the notation represents scalar variables.*

Symbol	Description
t_0	Current time-step
$\mathbf{P}_a(t)$	Position of agent a at time t (2D point)
$\mathbf{p}_a$	Current position of agent a ($\mathbf{p}_a = \mathbf{P}_a(t_0)$)
$\mathbf{v}_a$	Velocity of agent a (2D vector)
s_a	Speed of agent a ($s_a = \|\mathbf{v}_a\|$)
$s_{a_{comf}}$	Comfort speed of agent a
θ_a	Orientation of agent a (angle measured with the x-axis)
α_g	Bearing angle with respect to the goal
$\mathbf{P}_{o_i}(t)$	Position of obstacle o_i at time t (2D point)
$\mathbf{p}_{o_i}$	Current position of obstacle o_i ($\mathbf{p}_{o_i} = \mathbf{P}_{o_i}(t_0)$)
$\mathbf{v}_{o_i}$	Velocity of obstacle o_i (2D vector)
$\mathbf{v}_{o_i\|a}$	Velocity of obstacle o_i relative to agent a (2D vector)
$\mathbf{p}_{o_i\|a}$	Current position of obstacle o_i relative to agent a
$ttca_{o_i,a}$	Time to closest approach between agent a and obstacle o_i
$dca_{o_i,a}$	Distance between agent a and obstacle o_i at the closest approach

The current state of an agent a is defined by its position $\mathbf{p}_a$, orientation θ_a, and a velocity vector $\mathbf{v}_a$ given by:

$$\mathbf{v}_a = (v_{xa}, v_{ya}) = (s_a \cos\theta_a, s_a \sin\theta_a) = s_a \hat{\mathbf{v}}_a \ . \qquad (1)$$

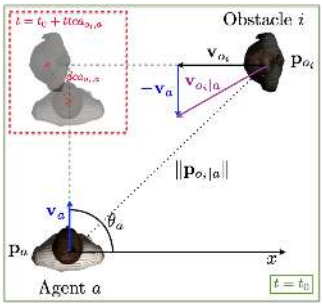


Figure 3: *Illustration of the variables used to model interactions.*

These quantities and others which are used throughout the paper are summarized in Table 1 and illustrated in Fig. 3. A more detailed description of these quantities is given in the following sections.

4. Perception: Acquiring information

The *perception* phase (see top left of Fig. 2) is the phase where each agent gathers information about the surrounding environment. In our implementation, this step is achieved by equipping agents with a synthetic vision, as in [OPOD10]. This technique results in a specific way to model and abstract obstacles, which are sampled and represented as a cloud of points moving relatively to the agent. As described in [OPOD10], this is achieved by graphically rendering the scene from the agent's point of view. Each visible obstacle is then perceived as one or more pixels, the set of which composes the perceptual space $\mathcal{O} = \{o_i\}$. In other words, $\mathcal{O}$ is the cloud of points representing obstacles and independently moving relatively to the agent. However, other more classical techniques for acquiring information, such as resorting to the position and shape of the obstacles typically known by the environment, could also be used.

For each point o_i perceived by a given agent, we compute its relative position $\mathbf{p}_{o_i|a}$ and relative velocity $\mathbf{v}_{o_i|a}$ as follows (see Table 1 for notations):

$$\mathbf{p}_{o_i|a} = \mathbf{p}_{o_i} - \mathbf{p}_a \ , \quad \text{and} \quad \mathbf{v}_{o_i|a} = \mathbf{v}_{o_i} - \mathbf{v}_a \ . \qquad (2)$$

$\mathbf{p}_{o_i|a}$ and $\mathbf{v}_{o_i|a}$ allow us to deduce $ttca_{o_i,a}$ and $dca_{o_i,a}$ (Fig. 1 (c) and Fig. 1 (d), respectively). Assuming that the relative motion of a point is constant, $ttca_{o_i,a}$ quantifies the remaining time before agent a reaches the minimum distance to the obstacle o_i; $dca_{o_i,a}$ is the distance between the agent and the obstacle at the time of the closest approach. They are computed as follows:

$$ttca_{o_i,a} = \begin{cases} t \in \mathbb{R} & : \mathbf{v}_{o_i|a} = (0,0) \\ -\frac{\mathbf{p}_{o_i|a} \cdot \mathbf{v}_{o_i|a}}{\|\mathbf{v}_{o_i|a}\|^2} & : \mathbf{v}_{o_i|a} \neq (0,0) \end{cases} , \qquad (3)$$

$$dca_{o_i,a} = \|\mathbf{dca}_{o_i,a}\| = \|\mathbf{p}_{o_i|a} + ttca_{o_i,a}\mathbf{v}_{o_i|a}\| \ , \qquad (4)$$

where t is any time value belonging to $\mathbb{R}$. For more details on the computation of $ttca_{o_i,a}$ and $dca_{o_i,a}$, please refer to the additional material document.

In Figure 1, the maps are encoded using a color code in which blue represents a value close to zero and red represents a large

value. Because we assume a linear motion and the agent is oriented parallel to the right wall (see Fig. 1 (a)), the *dca* of the agent with respect to the right wall is constant. The corresponding *ttca*, on the other hand, increases with the distance from a point on the wall to the agent. Note that Several Neuroscience articles describe the internal model humans use to anticipate obstacles' motion (e.g., [Kaw99]). It appears that linear extrapolation models often fit well empirical data. In the collision avoidance case, it was proven that linear extrapolation of obstacles' motion (in our case, points) is correlated with the presence of avoidance maneuvers. This means that the future *dca* value as we compute is relevant to encode the notion of risk of collision. The *ttca* and *dca* maps, along with the quantities defining an agent's motion (i.e., the speed, s_a; the orientation, θ_a; and the angle with respect to the goal, α_g) allow a full characterization of the current situation. This information is thus passed to the *evaluation* phase, which is described in detail in the next section.

5. Evaluation: A cost function to evaluate risk of collision

The goal of the *evaluation* phase (top right of Fig. 2) is to estimate the risk of collision with obstacles while the agent is steadfastly heading toward the goal. To this end, for each agent a, we define a cost function C_t composed of two terms:

$$C_t = C_m + C_o \,, \tag{5}$$

where the movement cost C_m accounts for whether or not the agent is heading toward the goal; and the obstacles cost C_o evaluates the importance of risk of collision with obstacles. In the following sections, we give the details regarding C_m and C_o.

5.1. Movement cost

The movement cost function C_m is defined so that it is minimal when the agent is heading toward the goal at its comfort speed:

$$C_m = 1 - \frac{C_\alpha + C_s}{2} \,, \tag{6}$$

where

$$C_\alpha = e^{-\frac{1}{2}\left(\frac{\alpha_g}{\sigma_{\alpha_g}}\right)^2} \,, \quad \text{and} \quad C_s = e^{-\frac{1}{2}\left(\frac{s_a - s_{a_{comf}}}{\sigma_s}\right)^2} \,. \tag{7}$$

α_g and s_a are the function arguments and σ_{α_g} and σ_s are parameters used to adapt the shape of the cost function. C_m is thus defined as a sum of two Gaussians (Eqs. (6) and (7)). The width of both Gaussians can be independently controlled through σ_{α_g} and σ_s. As detailed in Section 7.2, changing these parameters will directly play on the agents' avoidance strategy, i.e., their preference to adapt their speeds or their orientations to perform collision avoidance.

5.2. Obstacles cost

The cost $C_{o_i,a}$ accounts for the risk of collision between agent a and each perceived obstacle o_i. This function results from our interpretation that the risk of collision must be based on a notion of both distance (*dca*) and time (*ttca*). Considering only *dca* would make an agent react to objects on a collision course even though they are at very far distances. In the other hand, considering only *ttca* would make an agent react to obstacles in front of it moving in the same direction at very close distances because in this case *ttca* is infinite. Given that, the resulting cost of this function is high when both the distance at the closest approach *dca* (which indicates an existing risk of collision) and the time to closest approach *ttca* (which indicates how imminent the collision is) have low values. $C_{o_i,a}$ is defined as a two-dimensional Gaussian function:

$$C_{o_i,a} = e^{-\frac{1}{2}\left(\left(\frac{ttca_{o_i,a}}{\sigma_{ttca}}\right)^2 + \left(\frac{dca_{o_i,a}}{\sigma_{dca}}\right)^2\right)} \,, \tag{8}$$

where $ttca_{o_i,a}$ and $dca_{o_i,a}$ are the function arguments and σ_{ttca} and σ_{dca} are parameters used to adapt the shape of the function. σ_{ttca} controls avoidance anticipation time, whereas σ_{dca} controls the distance the agent must keep from obstacles.

Let us recall that the obstacle o_i corresponds to the obstacle seen by agent a through the pixel p_i (see Section 4). However, agent a can detect several obstacles from its visual flow (one for each pixel), where visual flow concerns the sequence of rendered images as seen by the agent. We combine the costs $C_{o_i,a}$ of each obstacle by simply averaging the cost of all the visible obstacles, to obtain the obstacles cost C_o as $C_o = \frac{1}{n}\sum_{i=1}^{n} C_{o_i,a}$, where n is the number of visible obstacles. Fig. 1 (e) shows the obstacles cost for the situation depicted in Fig. 1 (a). Note that the pixels for which the cost is high (marked in red in Fig. 1 (e)) correspond to those which have a low value for both *ttca* and *dca* (i.e., marked in blue in Fig. 1 (c) and (d)).

6. Action: Gradient descent

Each agent aims at moving toward its goal while avoiding some risk of collision with obstacles. As the cost function C_t models these two criteria, agents continuously adapt their velocity in order to locally optimize the cost function C_t. Technically, this is efficiently done by computing the gradient of C_t and by updating the agent's velocity to follow the gradient (steepest descent). This operation is repeated at each time-step.

We assume that an obstacle's motion is constant. The gradient of the cost function ∇C_t only depends on the agent's motion variables (s_a, θ_a). The agent's new motion, given by $(s_a^{(\text{new})}, \theta_a^{(\text{new})})$, is thus computed by giving a step of size λ_n such that:

$$(s_a^{(\text{new})}, \theta_a^{(\text{new})}) = (s_a, \theta_a) - \lambda_n \nabla C_t(s_a, \theta_a) + (0, \varepsilon) \,, \tag{9}$$

where a small noise value $\varepsilon \sim \mathcal{U}(-0.1, 0.1)$ is added to disrupt symmetric situations. Regarding the step λ_n used in the gradient descent, we have experimentally found that using $\lambda_n = 1$ yields good results. ∇C_t is evaluated as follows:

$$\nabla C_t = \left[\frac{\partial C_t}{\partial s_a} \; \frac{\partial C_t}{\partial \theta_a}\right] = \left[\left(\frac{\partial C_m}{\partial s_a} + \frac{\partial C_o}{\partial s_a}\right) \quad \left(\frac{\partial C_m}{\partial \theta_a} + \frac{\partial C_o}{\partial \theta_a}\right)\right] \,. \tag{10}$$

Note that the partial derivatives of both C_m and C_o can be explicitly evaluated as detailed in the additional material. The values of the partial derivatives of the obstacles cost function $C_{o_i,a}$, for the situation in Fig. 1 (a), can be visualized in Fig. 1 (f) and (g). In those images, blue represents a negative value, green a value close to zero, and red a positive value. Since most of the obstacles in Fig. 1 (f) with non-zero value have a blue color, the agent will tend to turn

left. In Figure 1 (g) the obstacles with non-zero (neutral) value have mostly a red color, causing the agent to reduce its speed.

7. Implementation and Parameters

7.1. Implementation

Algorithm 1 Gradient-based model implementation.

1: **for all** agents a **do**
2: $\mathbf{p}_a, \mathbf{v}_a \leftarrow$ get_state(a)
3: $camera \leftarrow$ set_up_camera$(\mathbf{p}_a, \mathbf{v}_a)$
4: $perc\_space \leftarrow$ render_environment$()$
5: **for all** pixels $p_i \in perc\_space$ **do**
6: **if** has_visible_obstacle$(\mathrm{p_i})$ **then**
7: $o_i \leftarrow get\_obstacle(p_i)$
8: $\mathbf{p}_{o_i}, \mathbf{v}_{o_i} \leftarrow$ get_motion(o_i)
9: $\mathbf{p}_{o_i|a}, \mathbf{v}_{o_i|a} \leftarrow$ relative_motion$(\mathbf{p}_a, \mathbf{v}_a, \mathbf{p}_{o_i}, \mathbf{v}_{o_i})$
10: $ttca_{o_i,a} \leftarrow$ compute_ttca$\left(\mathbf{p}_{o_i|a}, \mathbf{v}_{o_i|a}\right)$
11: $dca_{o_i,a} \leftarrow$ compute_dca$\left(\mathbf{p}_{o_i|a}, \mathbf{v}_{o_i|a}, ttca_{o_i,a}\right)$
12: $\left(\frac{\partial C_{o_i,a}}{\partial s_a}, \frac{\partial C_{o_i,a}}{\partial \theta_a}\right) \leftarrow$ grad_pixel_cost$(ttca_{o_i,a}, dca_{o_i,a})$
13: $perc\_space(p_i) \leftarrow \left(\frac{\partial C_{o_i,a}}{\partial s_a}, \frac{\partial C_{o_i,a}}{\partial \theta_a}\right)$
14: **end if**
15: **end for**
16: $\nabla C_o = \left(\frac{\partial C_o}{\partial s_a}, \frac{\partial C_o}{\partial \theta_a}\right) \leftarrow$ grad_obstacle_cost$(perc_space)$
17: $\nabla C_m = \left(\frac{\partial C_m}{\partial s_a}, \frac{\partial C_m}{\partial \theta_a}\right) \leftarrow$ grad_movement_cost(Δ_s, α_g)
18: $\nabla C_{t,a} \leftarrow$ grad_cost$(\nabla C_o, \nabla C_m)$
19: **end for**
20: **for all** agents a **do**
21: $\mathbf{v}_a \leftarrow$ adapt_agent_motion$(\nabla C_{t,a})$
22: **end for**

Algorithm 1 shows a pseudocode of our approach. It consists of two successive loops: computing C_t and its gradient $\nabla C_{t,a}$ for each agent a (lines 1 to 19); and a second loop to update the state of the agents in terms of $\nabla C_{t,a}$ (lines 20 to 22). The first loop fetches the agent's current position $\mathbf{p}_a$ and the current velocity $\mathbf{v}_a$ (line 2) which are then used to set up its virtual camera (line 3). In line 4, environment obstacles are rendered to the perceptual space (a texture). The loop over perceived pixels starts in line 5. If an obstacle o_i is visible through pixel p_i (line 6), we retrieve the corresponding obstacle and deduce its relative motion (lines 7 to 9). Note that the velocity $\mathbf{v}_{o_i}$ is not extracted from any analysis of the visual flow. Instead, we directly retrieve the velocity of visible 3D obstacles from the simulation state. To compute the velocity of each point, we use GPU interpolation. This approach allows treating objects of arbitrary shapes where different parts of the object move with different velocities (e.g. rotating volumes). In line 10, $ttca_{o_i,a}$ is computed according to Eq. (3). Similarly, in line 11, Eq. (4) is used to compute $dca_{o_i,a}$. The partial derivatives of the current obstacles cost are then computed in line 12. Next, the gradient of the obstacles cost (line 16), the gradient of the movement cost (line 17) and the gradient of the total cost (line 18) are computed. The second loop (lines 20 to 22) updates the simulation by iterating and adapting the motion of each agent, using the gradient ∇C_t and Eq. (9). Per pixel operations in lines 5 to 15 can be executed in parallel: they were implemented in OpenGL Shading Language (GLSL). More precisely, lines 7 to 9 were implemented in the vertex shader while lines 10 to 13 were implemented in the fragment shader. The computation of the gradient of the obstacles cost (line 16) is suitable for parallelization, but for now we have implemented just a CPU version. The rest of the algorithm was implemented using C++. As for the camera setup we have used the same settings as [OPOD10]: a field of view of 150°, a height of 80°, an orientation towards the ground with an angle of $-40°$, and a resolution of the vision texture of 256×48 pixels. The agents have been represented in a similar fashion to [OPOD10] so as to enable comparisons: cones with 0.3m of radius and 1.7m of height. However, our approach allows agents and obstacles to have arbitrary shapes and sizes.

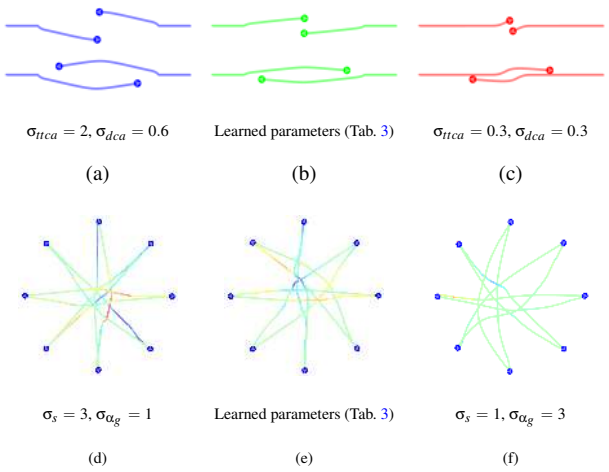


Figure 4: *Influence of the parameters on the agents' motion.* ***Top****: variation of the obstacles cost function's parameters.* ***Bottom****: variation of the goal cost function's parameters.*

7.2. Influence of model parameters

Our algorithm has four parameters: σ_{ttca} and σ_{dca} for the obstacles cost function; and σ_s and σ_{α_g} for the movement cost function. Their values can be intuitively adapted by users. Indeed, σ_{ttca} directly affects the agent's anticipation time whereas σ_{dca} modulates the minimum distance to obstacles. Their effects on avoidance trajectories are shown in Fig. 4 (a), (b) and (c). Increasing σ_{dca} results in a larger minimum distance between the agents whereas increasing σ_{ttca} results in an earlier motion adaptation to avoid collisions. In addition, movement cost parameters have direct impact on avoidance strategies. Greater σ_s values result in a preference for adapting speed in order to avoid collisions, whereas larger σ_{α_g} values favor the agents' reorientation. This effect is illustrated in the bottom line of Fig. 4: eight agents are initially disposed in a circle and have to reach the diametrically opposite position. The left plot (Fig. 4 (d)) corresponds to a large σ_s and a small σ_{α_g} (respectively 3 and 1): speed adaptation is favored over changes of direction. Trajectories are mainly rectilinear, some agents wait and give way to others rather than going around the obstacles (speed is color coded). The right plot (Fig. 4 (f)) corresponds to the opposite

configuration: a large σ_{α_g} and a small σ_s. It can be seen from the depicted trajectories that agents tend to prefer an adaptation of the motion's direction. Fig. 4 (e) shows results for an intermediate configuration (using the learned parameters of Table 3 discussed in the next section), in which the agents adapt both speed and direction to avoid collisions.

8. Results and Evaluation

In this section we provide results for both quantitative and qualitative analysis of the trajectories generated by our technique, as well as those generated by some previous models, under different scenarios. An in-depth discussion of the results presented in this section will then be made in Section 9.

8.1. Experimental set up

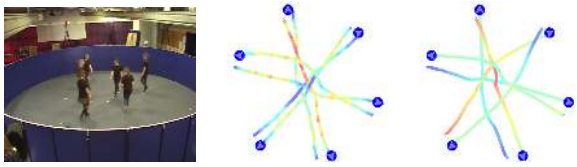
We evaluated our new approach by considering several different scenarios. We describe our results through the following sections. The companion video shows the corresponding animations with moving 3D characters and the provided additional material shows some variations of the proposed scenarios to demonstrate the robustness of the presented results.

Our model is compared with the following models: **OSV**, the original approach based on synthetic vision proposed by Ondřej et al. [OPOD10], which we selected as a representative of previous vision-based algorithms; **RVO2**, a broadly used algorithm (and therefore an excellent benchmark model), which is representative of velocity-based avoidance techniques [vdBGLM11], and; **PowerLaw**, the most recent state-of-the art technique [KSG14] evaluated against experimental data. For all these techniques we have used their standard formulations as provided by the authors. For example, as regards RVO2, we have only changed two parameters (radius and comfort speed) to make them adequate to our agents properties (0.5m and 1.5m/s, respectively).

We have also compared our model with a particle-based model [HFV00]. However, since its results were consistently worse than those of OSV, RVO2 and PowerLaw, we have not reported results. Ideally, we would compare to many other techniques, but we however retained the most relevant techniques to compare with. We have used two sets of parameters for our model in the tested scenarios which only differ slightly between each others (see Tab. 2). For the other models, we have used the default parameters (except for the comparison to real data, where optimal parameters were used for all tested models).

Table 2: *The two sets of parameters used for our model. Set 1 can be considered as a default parameter, since it is used for the most conventional scenarios. Set 2 is used for cases in which agents must perform drastic adaptation of speed.*

	σ_{ttca}	σ_{dca}	σ_{α}	σ_s
Set 1	1.0	0.3	2.0	2.0
Set 2	2.0	0.3	2.0	3.0



(a) Photo of the experiment (b) Experimental trajectories (c) Our model trajectories

Figure 5: *Comparison of experimental trajectories (b) with the trajectories generated by our model after fitting the model to data (c). It can be seen that, after calibration, the agents are able to follow the experimental data. Nevertheless, for some particular cases, a different parameter setting might be required.*

Table 3: *Model's parameters learned from data. N.D. stands for neighbor distance, M.A. for maximum acceleration, and M.N. for maximum number of neighbors.*

Our Model	OSV	RVO2	PowerLaw
$\sigma_{\alpha_g} = 2.0$	$A = 1.1$	$M.N. = 11$	$m = 2.4, k = 1.5$
$\sigma_s = 3.3$	$B = 1.6$	$N.D. = 16.2$	$N.D. = 11.6$
$\sigma_{ttca} = 1.8$	$C = 2.3$	$Agentt.h. = 5.6$	$\tau_0 = 4.47, ksi = 0.91$
$\sigma_{dca} = 0.3$	$s_{comf} = 1.6$	$Obstaclet.h. = 4.9$	$M.A. = 23.0$
$s_{comf} = 1.5$		$s_{comf} = 1.4$	$s_{comf} = 1.5$
			$ksi = 0.91$

8.2. Quantitative evaluation

8.2.1. Microscopic features

We have evaluated our model by comparing its generated individual trajectories and those generated by OSV, RVO2 and PowerLaw against real data. To this end, we have used the framework developed by [WGO*14] to learn the model's parameters from experimental data. We used motion captured data illustrated in Fig. 5(a) which, in our opinion, exhibit relatively rich behaviors with several agents interacting at the same time, hence providing interesting elements of comparison. Real and simulated trajectories were quantitatively compared using a *progressive difference* metric. This variant of Guy's metric [GvdBL*12] is prone to capture global motion features and avoids overfitting problems. Tab. 3 shows the values of the model's parameters after calibration. The results depicted in Fig. 5 (b) and (c) show that, after calibration, our model produces trajectories with motion adaptations similar to the real data. We also fit OSV, RVO2 and Power Law in the same way. The progressive difference metric resulted in the following scores: 0.26 for RVO2, 0.35 for OSV, 0.09 for Power Law and 0.06 for our gradient-based model, where a lower score means better performance.

Figure 6 shows the results of the four tested models when solving a classic circle scenario, where 20 agents must reach a diametrically opposed position. To break the symmetry, the initial agents' positions contain a small noise value. Such a scenario is used to test the models' ability to solve situations of potential agglomeration, in which different agents should take different decisions so as to reach their goal without collisions and without the emergence of

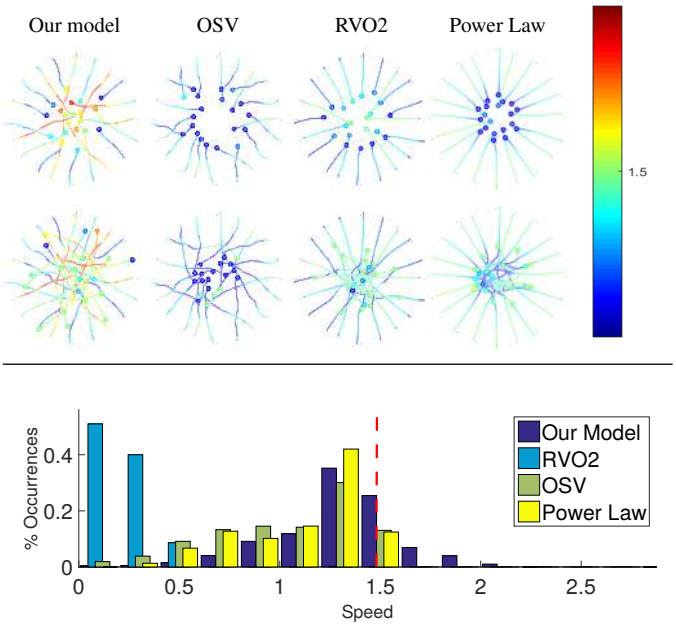


Figure 6: *Results for Circle scenario. Top: the agent's color encodes the speed. Blue means slow, green means comfort speed, and red means fast. Note that, our model is the only model which makes agents significantly accelerate to avoid collisions (although very tiny accelerations can seldom be remarked for the Power Law model). Bottom: histogram of the agent's speed, with the dashed red line marking their comfort speed.*

strong motion patterns. Fig. 6 (top) shows the resulting trajectories, at two different steps of the simulation for each model. The trajectories' color encodes the speed of the agent according to the color bar shown at the right: green for comfort speed (1.5 m/s); warm colors (yellow, orange and red) for speeds greater than the comfort speed; and cold colors (blue) for agents moving slower than the comfort speed. The histogram of Fig. 6 (bottom) shows the number of occurrences of the agents' speed during the simulation, for the four tested models. Note that RVO2 makes most of the agents move at a very low speed, and that our model is the one where more agents move at their comfort speed. Moreover, our model is the single one in which agents move faster than their comfort speed.

8.2.2. Macroscopic features

The objective of the analysis made in this section is to quantify and compare global features of the trajectories generated by the four tested models. Let us focus on the initial configuration of the *Crossing* scenario shown in Fig. 7 (top left), where two perpendicular flows of agents must cross. The trajectories generated using this initial configuration by our model, OSV, RVO2 and PowerLaw are shown from the second to the fifth columns of Fig. 7 (top). A histogram of the crossing distance between agents, shown in Fig. 7 (bottom left), characterizes the crossing distances for each model. Note that the occurrences of large distances is dominated by the OSV model and the occurrences of distances very close to zero are dominated by RVO2 and PowerLaw. Moreover, when analyzing real crowds crossing in similar configurations, recent literature has reported the emergence of 45° lanes patterns on the crowd structure after the two flows cross [CARH13]. We have quantitatively evaluated this effect by implementing the clusters detection algorithm of [MGM*12] which states that "two pedestrians belong to the same cluster at a given moment of time if one of them is following the other". We determine the orientation of each detected cluster by fitting a line to its agents. The bottom row of Fig. 7 shows the detected clusters for a given step in the crossing scenario. Note that no clusters have been detected for RVO2 and PowerLaw, and that the average slope of the clusters found is of 45° for our model, and 41° for OSV.

8.3. Qualitative evaluation

Synthetic vision enables processing static and dynamic obstacles of any shape in the same fashion: everything is abstracted as visible pixels moving relatively to each agent. This process also implicitly filters invisible obstacles and gives priority to nearer obstacles (of equivalent size) as they are represented by more pixels. These properties are illustrated by the *Room*, *Columns* and *H-corridor* scenarios which mix static obstacles and moving agents. For more detailed results on each of the scenarios, please refer to the accompanying video or to the additional material.

Another non-trivial scenario is given by Fig. 8, where two groups of agents with opposite flow directions must cross, while avoiding a set of static obstacles (columns). The presence and position of such obstacles forces the groups to reorganize and break their initial formation to successfully reach their goal. Our model and OSV accomplish this goal although with rather different trajectories. On the other hand, RVO2 and PowerLaw cause part of the agents to stay stuck behind the obstacles. The histogram at the left gives an insight on the agents' speed in this scenario. Note that for our model the large majority of the occurrences corresponds to agents moving close to their comfort speed ($1.5 m/s$).

In the *H-corridor* scenario (Fig. 9), a group of controlled agents (in red) has the goal of horizontally traversing a corridor from left to right. However, two groups of non-controlled agents (in green) vertically cross the corridor, from the top to the bottom, temporarily obstructing the controlled agents' path. To reach their goal without collisions, some of the agents must stop, wait for the moving obstacles to pass, and finally restart their movement towards the goal. Fig. 9 shows that the four models provide clearly different results: our model perfecly solves the scenario; agents controlled by OSV can, with a single exception, stop to avoid collisions and then retake their movement; for RVO2 some of the agents are dragged by the non-controlled agents; and finally, for PowerLaw, agents tend to agglomerate after the passage of the non-controlled agents.

8.4. Performance

Simulations ran on a MacBook, 2.13 GHz Intel Core 2 Duo processor, 4.0 GB of RAM, NVidia GeForce 9400M graphics card, 16 CUDA cores and 256 MB of dedicated RAM. The algorithm was written in C/C++ and we used OpenGL for graphics. Fig. 10 displays the simulation loop runtime compared with OSV on the *Op-*

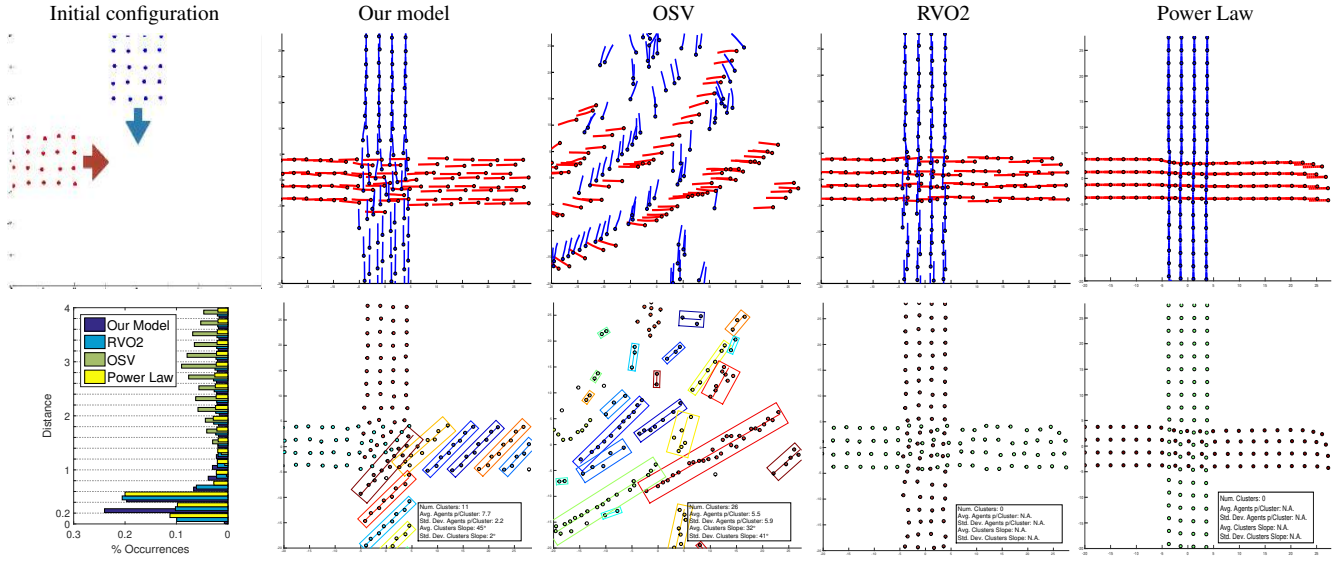


Figure 7: *Results for the* Crossing *scenario. The image at top-left shows the initial configuration of the scenario. The images in the following columns show the groups of agents separated by flow (red goes to the right and blue goes to the bottom) on the top and separated by clusters on the bottom for each of the three tested models. The image at bottom-left shows the distribution of distances for each model.*

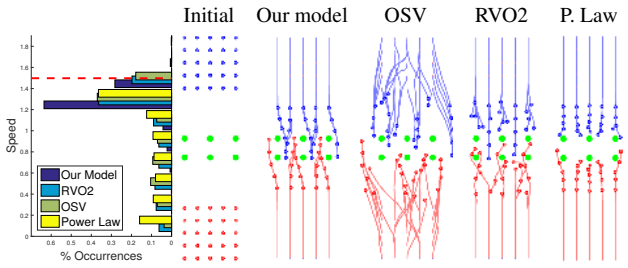


Figure 8: *Results for the* Columns *scenario. The histogram on the left shows the distribution of speeds for each model, with the dashed red line marking comfort speed. The remaining figures show a comparison of the trajectories synthesized by the four tested models.*

posite scenario. Synthetic vision techniques cannot compete with RVO2-like geometrical approaches, but still show reasonable performance, being able to run at 10 FPS for 300 agents. Compared with OSV, we perform a larger amount of computations in GPU. However, we only need a 2-channel texture to store the results (the two partial derivatives), whereas OSV requires a 4-channel texture. This makes the GPU/CPU data transfer time slightly faster for our model. Moreover, we only process two texture channels in CPU whereas OSV must process four. This is why, overall, both methods roughly exhibit the same execution time.

9. Discussion

9.1. Comparison with previous approaches

In this section, we discuss the results presented in the previous section with a view on comparing the four tested models. Note that these four models work in the velocity space and on *dca* and *ttca* values to allow agents anticipating their reactions. Their difference lies in the guiding principles of each approach according to which agents are steered. RVO2 steers agents so as to always keep *dca* over a threshold value when the time horizon is bounded. OSV takes a similar approach but, however, *dca* is not explicitly computed. Instead it is estimated from the motion of pixels in the virtual optical flow. PowerLaw will steer agents to always increase *ttca* values. Finally, in comparison, our method will steer agents so as to always reduce the risk of future collision, by penalizing low pairs of *dca* and *ttca* values.

We first compared the *avoidance strategies*. Our algorithm shows a very heterogeneous strategy to avoid collisions, with the agents exploring the full range of available adaptations (any combination of acceleration, deceleration or direction adaptations), as it was observed for real humans performing collision avoidance [OMC*13]. This behavior is clearly visible in Fig. 6. Trajectory traces exhibit early re-orientations, and the speed histogram demonstrates that our method covers a larger spectrum of attainable speeds. In comparison, OSV agents are limited to a small set of reactions (turn to avoid future collisions, and decelerate to avoid imminent collision). RVO2 and PowerLaw penalize large deviations from the desired velocity vector (pointing towards the goal), and thus do not explore the full range of possible strategies. For example, RVO2 will always retain a solution velocity at the boundary of the constraints imposed by a moving obstacle, at the closest distance from the desired velocity: strategies of the same kind are

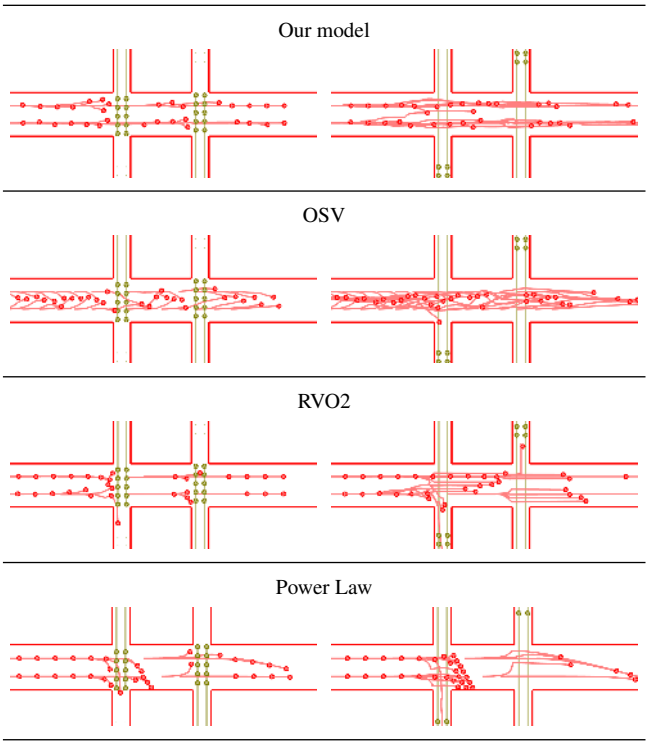


Figure 9: *Comparison of the trajectories synthesized by the four tested models for the* H-corridor *scenario.*

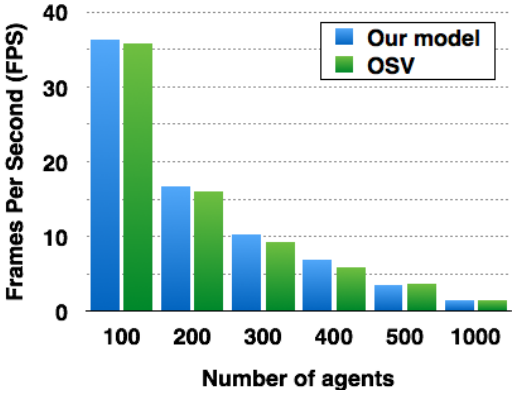


Figure 10: *Performance comparison for the Opposite scenario with different number of agents.*

thus repeatedly selected. The quantitative comparison of the various simulations against real data (Section 8.2) proves that, in the selected example, the variety of strategies employed by our model leads to trajectories which are closer to the real data. This variety and their impact on the final results can also be observed in the companion video.

We also compared our results in terms of *interaction distances* between agents. It is logical to assume that humans accept close interaction distances in dense situations, but may take larger ones when room is available around them. Our method results in such types of adaptation, because agents tend to always diminish the risk of future collision even when the future distance of closest approach is greater than contact distance. This can be observed in the H-corridor scenario (Fig. 9): interaction with the groups moving vertically makes the controlled agents (in red) get closer to each others, but a relaxation occurs after interaction and the red agents recover their initial interpersonal distances. Such relaxation is not observed for RVO2. Indeed, after interaction, the absence of future collision will make agents keep moving straight. The PowerLaw approach behaves in a similar way. Finally, OSV tends to make agents move in the same direction, following each other. This emergence of a following behavior occurs because the predecessor agent occludes other obstacles while not provoking a reaction of the follower. The distance histogram in the crossing scenario (Fig. 7) also reveals the difference of distance distributions resulting from the considered algorithms. Note that this distance adaptation does not prevent our method from exploring narrow paths. When a collision risk exists everywhere, such as in the Columns scenario (Fig. 8), the local optimization loop is able to find the minimum in the cost function and to steer agents in this difficult situation. In the Columns scenario, an efficient self-organization of the traffic can be observed. Moreover, for scenarios where physical obstacles are present (such as the Columns scenario and the H-corridor), the improved performance of our model is probably better explained not only from better agent-agent interaction, but also from better agent-obstacle collision avoidance. Indeed, the models RVO2 and Power Law focus on agent-agent interactions and assume rather simplistic obstacle representations, which could explain some behavioral issues observed in these scenarios.

The situation introduced in Fig. 7 demonstrates the ability of our approach to produce the formation of 45° bands (acknowledged by the literature [CARH13]) when orthogonally moving flows meet. OSV is also able to achieve it, however, the tendency of OSV agents to turn to avoid future collisions (see above) results in large deviations from the initial path, that we estimate to be exaggerated. RVO2 and PowerLaw are not able to simulate this emergent pattern: the reason is again that these algorithms restrict agents to the smallest required adaptation to avoid collision (agents adapt to pass at contact distance).

9.2. Limitations

Despite the results obtained, some limitations of our approach can be pointed out. Firstly, a quantitative evaluation of how close our control loop is to one of real humans is not possible at the current stage. This is because the definition of real human's control loop in crowds is still an open problem and thus there are no available metrics to perform such an evaluation. Secondly, our cost function is limited to collision avoidance and goal reaching behaviors. The extension to other behaviors is discussed below. Thirdly, our method is not able to deal with complex navigation in large environments and should be integrated to a global motion planner. Finally, as the notion of risk of future collision exists beyond the contact threshold, agents may perform maneuvers the origin of which is sometimes difficult to interpret.

Another limitation concerns the parameter setting. Although we

have kept the parameters setting intuitive and the number of parameters low, we could not find a single parameter setting which fits all types of situations. More precisely, on the one hand, we could find a set of parameter values that fit a similar context, such as the *Opposite*, the *Columns* and the *Crossing* crossing examples which show similar situations of intersecting flows of agents moving in two different directions. On the other hand, we had to adapt parameters values for different contexts. In the *H-Corridor* the situation, adapting the speed obviously is a more efficient strategy than adapting the direction, due to the presence of walls on the side. Consequently, we had to increase $\sigma_{s_{comf}}$ to favor this strategy. Note that parameter sensitivity to context is a more general limitation for all existing algorithms.

9.3. Possibilities of extending to a geometrical approach

In our work, we equip agents with a synthetic vision system to perceive the surrounding environment. This choice has several interesting properties. One of them is an implicit model which weights the interactions between agents and obstacles. Indeed, the sampling density of each visible obstacle depends on the position of the obstacle relatively to the agent: the closer the obstacle, the more pixels correspond to it, which causes the obstacle to have a larger weight in the cost function. Another interesting property is that visibility and partial occlusions are automatically accounted for (hidden and out-of-the-field parts of the obstacle are surely not sampled). Finally, abstracting the obstacles as sets of moving points simplifies the computation of $ttca$ and dca terms (Eq. (4)), whilst the real shape of obstacles can still be considered. Even though we use simplified shapes for agents, we consider the volume they occupy, and the approach would easily support agents and obstacles of more complex shapes.

Given the above, an interesting question is the possibility of applying the same control loop described in Section 3.1 to agents without synthetic vision (e.g., agents perceiving the surrounding environment through a classic geometrical modeling approach to identify neighbor agent in interaction). This is perfectly doable. In such a case two questions need to be addressed. First, a criterion should be defined to determine which agents or obstacles an agent should interact with. Second, the obstacles cost function needs revision so as to explicitly weight the obstacles according to some criteria such as, for example, the distance to the agent and the volume of the obstacle.

9.4. Extension to other behaviors

In our approach, agents are able to perform collision avoidance during goal directed navigation tasks. We believe that our control scheme can relatively easily be extended to other kinds of behaviors. The main principle governing an agent's motion is formulated as a cost function which is constantly locally optimized. A technical solution to extend other solutions would be to create a set of different cost functions, each one describing one kind of behavior. Then, we could imagine different approaches to successively select cost functions to chain different behaviors, or to mix them by weighting a combination of functions, so that agents move according to several simultaneous criteria. The limitation of this approach is that each behavior should be formulated as a cost function which is formally derivable with respect to orientation and speed.

10. Conclusion

In this paper, we present a new simulation algorithm based on synthetic vision. Our main contribution is a new steering scheme based on: a) a cost function which evaluates the agents' situation w.r.t. their target and risk of collision and b) a locally optimal gradient-descent scheme. Our results show that we can generate high-quality trajectories which is also a useful feature for entertainment applications. In particular, our new method prevents some known artifacts generated by previous methods as demonstrated in the previous section. Our contribution is not a simple revisiting of vision-based algorithms, it makes this method more robust, especially with its new mathematical foundations. The new control scheme is more easily extendable to new behaviors and types of interactions between agents by defining some corresponding cost functions. Besides, we changed the mechanism by which agents adapt their trajectory: our agents continuously perform locally optimal adaptations whereas previous approaches set logical criteria to trigger adaptations.

In our opinion, our model shows a more human-like behavior than the other models. We base our opinion on the fact that our model is able to more closely follow the experimental data than the other models (see Section 8.2.1). Moreover, our model produces agent behaviors which are commonly observed in the day-to-day life, but which the other models cannot reproduce, such as accelerating to avoid collisions. Finally, the presented histograms indicate that, in our model, agents keep a more natural distance between them, which reduces the strong and unrealistic patterns produced by other models.

Future work aims at directly tackling the limitations we identified in the previous section. Regarding the variation of parameters with different scenarios, we think that the focus of future research should be put on identifying the current situation and selecting the appropriate parameter setting, rather than trying to find a single parameter setting which would work well in all situations (and which might not exist). To this end, one could resort to machine learning and image processing techniques, so as to classify the situations and to learn their respective parameters based on features of the visual flow. The use of a data-driven layout design approach such as the one proposed in [FYY*16] could also be useful for helping defining/validating proper parameters for specific scenarios.

To extend our method with a richer set of behaviors, as well as to handle larger sets of contexts such as high density ones, we plan defining and combining different cost functions. As an example, instead of $ttca$ and dca, a cost function for 'high-density' contexts could be based on the pressure agents perceive from other agents in contact with them, or the simple distance they keep to the others. Another example is interactions with more dynamic obstacles like cars, for which a notion of collision energy could be added to avoid more carefully the most threatening obstacles. With such future extensions, we expect to develop more generally the idea of perceptual models for crowd simulation, and to provide the relevance of the subtleties of human behaviors in such contexts. Finally, the model could also be improved based on the observation that, in real life, we react differently when avoiding humans or other obstacles.

Indeed avoiding humans implies keeping an interpersonal distance which varies according several aspects. However, when avoiding other obstacles (e.g., a column or a wall), this interpersonal need not be respected. Future work could aim at modifying the obstacle cost function in order to treat differently those two types of entities.

Acknowledgements

T. B. Dutra acknowl1edges CAPES for the fellowship (PDSE Proc. 0130/13-3) and J. Ondřej for helping in the implementation of his model. R. Marques acknowledges the projects Percolation (ANR-13-JS02-0008), and Kristina (H2020-RIA-645012) for partially financing this research.

References

[ANMS13] ALI S., NISHINO K., MANOCHA D., SHAH M.:. In *Modeling, Simulation and Visual Analysis of Crowds: A Multidisciplinary Perspective*. Springer New York, 2013. 2

[CARH13] CIVIDINI J., APPERT-ROLLAND C., HILHORST H. J.: Diagonal patterns and chevron effect in intersecting traffic flows. *EPL (Europhysics Letters) 102*, 2 (2013), 20002. 8, 10

[Che04] CHENNEY S.: Flow tiles. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics symp. on Computer animation* (2004), pp. 233–242. 2

[CVB95] CUTTING J., VISHTON P., BRAREN P.: How we avoid collisions with stationary and moving objects. *Psychological Review 102*, 4 (1995), 627–651. 3

[FYY*16] FENG T., YU L.-F., YEUNG S.-K., YIN K., ZHOU K.: Crowd-driven mid-scale layout design. *ACM Trans. Graph. 35*, 4 (July 2016), 132:1–132:14. 11

[GCK*09] GUY S. J., CHHUGANI J., KIM C., SATISH N., LIN M., MANOCHA D., DUBEY P.: Clearpath: Highly parallel collision avoidance for multi-agent simulation. In *Proc. of the 2009 ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 177–187. 2

[GvdBL*12] GUY S. J., VAN DEN BERG J., LIU W., LAU R., LIN M. C., MANOCHA D.: A statistical similarity measure for aggregate crowd dynamics. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 190:1–190:11. 7

[HFV00] HELBING D., FARKAS I., VICSEK T.: Simulating dynamical features of escape panic. *Nature 407*, 6803 (2000), 487–490. 2, 3, 7

[HM95] HELBING D., MOLNÁR P.: Social force model for pedestrian dynamics. *Physical Review E 51*, 5 (1995), 4282–4286. 1, 2

[JCP*10] JU E., CHOI M., PARK M., LEE J., LEE K., TAKAHASHI S.: Morphable crowds. *ACM Trans. Graph. 29* (2010), 140:1–140:10. 2

[Kaw99] KAWATO M.: Internal models for motor control and trajectory planning. *Current opinion in neurobiology 9*, 6 (1999), 718–727. 5

[KHvBO09] KARAMOUZAS I., HEIL P., VAN BEEK P., OVERMARS M.: A predictive collision avoidance model for pedestrian simulation. In *Motion in Games*, vol. 5884 of *LNCS*. Springer, 2009, pp. 41–52. 2, 3

[KLLT08] KWON T., LEE K. H., LEE J., TAKAHASHI S.: Group motion editing. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 80:1–80:8. 2

[KSG14] KARAMOUZAS I., SKINNER B., GUY S. J.: Universal power law governing pedestrian interactions. *Phys. Rev. Lett. 113* (Dec 2014), 238701. 3, 7

[KSH*12] KAPADIA M., SINGH S., HEWLETT W., REINMAN G., FALOUTSOS P.: Parallelized egocentric fields for autonomous navigation. *The Visual Computer 28*, 12 (2012), 1209–1227. 2, 3

[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. 2, 3

[LCL07] LERNER A., CHRYSANTHOU Y., LISCHINSKI D.: Crowds by example. *Computer Graphics Forum 26*, 3 (2007), 655–664. 2

[MGM*12] MOUSSAÏD M., GUILLOT E. G., MOREAU M., FEHRENBACH J., CHABIRON O., LEMERCIER S., PETTRÉ J., APPERT-ROLLAND C., DEGOND P., THERAULAZ G.: Traffic instabilities in self-organized pedestrian crowds. *PLoS Comput Biol 8*, 3 (03 2012), e1002442. 8

[NGCL09] NARAIN R., GOLAS A., CURTIS S., LIN M. C.: Aggregate dynamics for dense crowd simulation. *ACM Transactions on Graphics 28* (2009), 122:1–122:8. 2

[OMC*13] OLIVIER A.-H., MARIN A., CRÉTUAL A., BERTHOZ A., PETTRÉ J.: Collision avoidance between two walkers: Role-dependent strategies. *Gait & posture 38*, 4 (2013), 751–756. 9

[OPOD10] ONDŘEJ J., PETTRÉ J., OLIVIER A.-H., DONIKIAN S.: A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graph. 29*, 4 (July 2010), 123:1–123:9. 3, 4, 6, 7

[PAB08] PELECHANO N., ALLBECK J. M., BADLER N. I.: Virtual crowds: Methods, simulation, and control. *Synthesis Lectures on Computer Graphics and Animation 3*, 1 (2008), 1–176. 2

[POO*09] PETTRÉ J., ONDŘEJ J., OLIVIER A.-H., CRETUAL A., DONIKIAN S.: Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proc. of the 2009 ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 189–198. 2

[PPD07] PARIS S., PETTRÉ J., DONIKIAN S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum 26*, 3 (2007), 665–674. 2

[PvdBC*11] PATIL S., VAN DEN BERG J., CURTIS S., LIN M. C., MANOCHA D.: Directing crowd simulations using navigation fields. *IEEE Transactions on Visualization and Computer Graphics 17* (February 2011), 244–254. 2

[Rey87] REYNOLDS C.: Flocks, herds and schools: a distributed behavioral model. *SIGGRAPH Comp. Graphics 21*, 4 (1987), 25–34. 2

[Rey99] REYNOLDS C.: Steering behaviors for autonomous characters. In *Game Developers Conference 1999* (1999), pp. 763–782. 2, 3

[RRW14] RIO K. W., RHEA C. K., WARREN W. H.: Follow the leader: Visual control of speed in pedestrian following. *Journal of Vision 14*, 2 (2014). 3

[SKH*11] SINGH S., KAPADIA M., HEWLETT B., REINMAN G., FALOUTSOS P.: A modular framework for adaptive agent-based steering. In *Symp. Int. 3D Graphics and Games* (2011), pp. 141–150. 2

[ST07] SHAO W., TERZOPOULOS D.: Autonomous pedestrians. *Graphical Models 69*, 5-6 (2007), 246–274. 2

[TCP06] TREUILLE A., COOPER S., POPOVIĆ Z.: Continuum crowds. In *SIGGRAPH '06* (2006), ACM, pp. 1160–1168. 1, 2, 3

[TM13] THALMANN D., MUSSE S.:. In *Crowd Simulation*. Springer London, 2013. 2

[vdBGLM11] VAN DEN BERG J., GUY S., LIN M., MANOCHA D.: Reciprocal n-body collision avoidance. In *Robotics Research*, vol. 70. Springer, 2011, pp. 3–19. 2, 7

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

[WF04] WARREN W., FAJEN B.: Optic flow and beyond. Kluwer Academic Publishers, Norwell, MA, USA, 2004, ch. From optic flow to laws of control, pp. 307–337. 3

[WGO*14] WOLINSKI D., GUY S., OLIVIER A.-H., LIN M., MANOCHA D., PETTRÉ J.: Parameter Estimation and Comparative Evaluation of Crowd Simulations. *Computer Graphics Forum 33*, 2 (2014), 303–312. 7