An Introduction to Crowd Simulation
Eurographics 2017 Tutorials Program
Course Notes

Julien Pettré, Inria
Nuria Pelechano, UPC

March 13, 2017

Contents

1 Lecturers **2**

2 Class Notes: Introduction to Crowd Simulation **3**

2.1 Velocity-based crowd simulation algorithms 3

2.1.1 Introduction 4

2.1.2 State of the art 5

2.1.3 Experimental validation 6

2.1.4 3 examples of velocity-based models 8

2.1.5 Conclusion 12

2.2 Animation of crowd characters 16

3 Annexes **44**

Part 1

Lecturers

- Julien Pettré, Inria, Rennes, France
  julien.pettre@inria.fr
- Nuria Pelechano, UPC, Barcelona, Spain
  npelechano@cs.upc.edu

Part 2

Class Notes: Introduction to Crowd Simulation

2.1 Velocity-based crowd simulation algorithms

A crowd is defined as a set of individual gathered in a same location because they share common goals. In a crowd, each individual has interactions with his neighbors. These interactions can be of many kind and can be influenced by a large set of factors which relate to individual properties or the milieu, such as for example physical, psychological, social or environmental factors. Crowds in motion are most often studied. Crowds moving in public places and buildings or social events are typical study cases. In moving crowds, interactions are of mostly physical: during their motion, individuals in crowds avoid or follow each other, group, disperse, etc. The combination of all these interactions result into typical large-scale emergent structures, which determine the main characteristics of crowd motions.

Simulating crowds is important for many reasons. Architects simulate crowds to make predictions about pedestrian traffic flows and to estimate the level of service of public buildings. Animation designers simulate crowds to make visually appealing scenes of battlefields, or to populate a virtual scenery. Game designers simulate crowds to make lively scenarios in virtual cities, etc. Two major classes of approaches can be distinguished. The macroscopic approaches directly consider the global aspects of a moving crowd like for example modeling it as a viscous fluid.

Microscopic crowd simulation enable generating continuous and smooth trajectories for individual agents. They are based on some local model interactions between agents. The global behavior of a crowd is then an emergent phenomenon resulting from the combination of the numerous interactions that agents have. A crowd simulator is then at least composed of the following elements:

1. a neighbor selection process: which determines which set of neighbor agents are interacting with a given agent,
2. a local model of interaction: which determines how each interaction influences agents motion,
3. an interaction combination process: which integrates all the interactions a given agent is undergoing

A lot of recent efforts in the graphics research has been dedicated to the design of new local models of interactions. This class explains the novelties introduced by *velocity-based models* that were introduced a tenth of years ago and which, in contrast with previous approaches, are able to simulate how humans drive their motion by anticipating others motion. Since their introduction, many developments have

built on this principle, which resulted into a breakthrough in the level of realism achieved by crowd simulators.

The first part of this class is dedicated to the definition of velocity-based local model of interactions. We also explain why those models enable agents performing anticipated reactions. In a second part, we present one of the most popular velocity-based crowd simulation algorithm, RVO, as well as a set of its variants. In a third part, we dedicate a large part of the class to applications. The Golaem company describes how velocity-based models enabled more realistic simulation for the design of train coaches or enhanced visual aspect of crowded scenes for movies. Unity3D describes how RVO benefits to entertainment applications.

2.1.1 Introduction

The main objective of microscopic crowd simulation is to compute the macroscopic behavior of a crowd by simulating the interactions people have together at a local scale. Pedestrian simulation is a typical example. We expect to prediction of global traffic conditions from numerical models of the physical interactions people have during their navigation. Basically, they avoid all the static and moving obstacles in their neighborhood: collision avoidance is generally considered to be the most crucial interaction for pedestrian crowd simulation, and the absence of interpenetration between simulated bodies to be a hard constraint.

Velocity-based models corresponds to a new type of numerical models of microscopic interactions for crowd simulation. They recently appeared in the literature, in 2007, and various solutions were developed from various fields: computer graphics, computational geometry, computer vision and cognitive science. These fields took interest in velocity-based models for various reasons, but a common one is the need for realistic simulation results, or at least believable ones, at all scales of the crowd, even at the smallest one. Too many artifacts were produced by previous techniques, especially because of their lack of anticipation: agents were often trapped in dead-lock situations or producing strange oscillating motions. Avoiding a collision with anticipation means that avoidance maneuvers are over before agents get close to each other. Anticipation cannot result from microscopic models that formulates interaction as a function of distance between agents (which is the case of most of simulators based on particle systems and physically inspired models).

To allow anticipation, a velocity-based model formulates interactions not only as a function of agents' states (positions), but also as a function of their derivatives (velocities). The basic principle of velocity-based models is to decompose, for each agent, the reachable velocity domain (all the global motions an agent can perform) into two components: the admissible, and the inadmissible velocity domains. The admissible velocity space is the set of velocities at which an agent can move without risk of future collision. At the opposite, a risk of collision appears when the agent moves at a velocity belonging to the inadmissible domain. Obviously, the notion of collision risk is considered with notion of time. Time-to-collision (TTC) is a classically used variable to describe the risk with respect to the time dimension.

How to compute the admissible and inadmissible velocity domains in the situation of a crowd, when all obstacles are constantly moving and each agent is performing adaptations independently? By working with short time-windows and by constantly updating agents' states and decisions, we show that velocity-based models provide convincing and smooth simulation results. Some recent efforts in models evaluation on real data showed that this category of models is promising for realistic crowd simulation. The objective of this paper is to provide an overview of some existing solutions. The paper is organized as follows: section 2 situates velocity-based models among other simulation techniques, and gives a list of existing velocity-based models, as exhaustive as possible. Based on experimental observations of collision avoidance behaviors, Section 3 shows that velocity-based models are grounded in reality.

Section 4 describes 3 existing velocity -models in the objective to provide a pedagogical description and to emphasize their singularities. Finally Section 5 further discusses differences between models and proposes future development paths for this new category of models.

2.1.2 State of the art

Crowd simulation can be approached from two opposing perspectives. One is to consider it as a coherent body, which is the macroscopic approach. The other is to build the crowd from local inter-agent interactions, which is the microscopic approach.

From the macroscopic point of view [13, 5], a crowd is modeled to behave like a fluid, thus allowing to use fluid dynamics inspired concepts such as velocity potential fields. The main focus of this type of approach is to obtain a coherent behavior from the crowd, leaving aside individual agents' goals and constraints and enforcing non-interpenetration (the primary constraint for crowd simulation) at the last moment. Since these approaches have been designed with only global patterns in mind, many artifacts appear at the local scale. It is for example possible to see agents moving sideways, collisions or even residual inter-penetrations.

On the contrary, from the microscopic point of view, individual constraints and goals are most important. A global behavior is then expected to emerge as a result of these complex, local interactions. Various methods exist to simulate these interactions.

A possibility is to discretize the space into cells as in cellular-automata and model agents as occupied cells [12]. In this case, interactions are modeled using (often probabilistic) transition rules and their complexity varies with the discretization. Different behaviors can be modeled this way such as the formation of lanes when two groups cross ways for example. However, due to the necessary levels of discretization, it becomes impractical to model complex collision avoidance strategies; this is usually handled by forbidding agents to move to an already occupied cell.

Non discrete models, on the other hand, let agents make decisions based on several different inputs and criteria. The first such model (Boids model) has been proposed in [11] where agents made decisions based on three rules: separation (agents avoid overcrowding), alignment (agents steer towards a common goal) and cohesion (agents keep close to the group). Another model has been proposed in [3, 4] as an analogy to physics where agents were subject to forces, hence the name Social Forces model. This was a position-based model, all forces which affected the agents were obtained based on their positions. This allowed to model more complex collision avoidance behaviors as well as other phenomena such as friends and store fronts where the forces would be attractive instead of repulsive as is the case for obstacles.

In the recent years, velocity-based models emerged as an evolution of the position-based Social Forces model. These models, also called predictive models, process more information than the position-based ones. They are essentially able to predict the trajectory of the agents and make decisions accordingly. The most popular way of doing so is to establish an admissible velocity domain which contains all velocities that will not lead to a collision. All that remains is then to choose the velocity that is closest to a preferred velocity (for example one that leads to the goal) thus necessitating the smallest acceleration.

The Dynamic Window Approach, proposed in [2], aimed to allow robots to avoid collisions. The method is essentially a greedy exploration algorithm of possible velocities weighted by a cost based on the robots' dynamics (the cost is very high if the velocity leads to a collision). This model later inspired the velocity-obstacle method [1]as well as the first collision avoidance approaches in the field of crowd simulation.

Paris presented a technique to perform collision avoidance reasoning in the velocity space in[8]. The principle is to divide the space in front of a given agent into sectors and for each of these, it establishes

a minimum and maximum speed which form the interval of speeds that lead to a collision. The agent is then free to choose the velocity outside these intervals that is closest to his preferred speed.

Following, van den Berg introduced the Reciprocal Velocity Obstacle (RVO) model [14]. In this approach, pairs of agents are represented in the relative velocity space and each velocity (a point in velocity space) that leads to a collision in a certain time window is made unavailable, thus forming a velocity obstacle. The chosen velocity is then the closest one to the preferred velocity and outside the obstacle. This model has since been updated with the ability to divide the avoidance effort between agents as well as the ability to use acceleration information (leading to the newer Acceleration-Velocity Obstacle model).

Proposed in [9], the Linear Trajectory Avoidance (LTA) model explores a set of possible moves and associates a cost to each of them. The cost is a function of future crossing distance, which is estimated by extrapolating trajectories given the current positions and velocities. The model was used in the framework of video tracking of real pedestrians. It acts as a predictor for the tracker, and was proved to significantly increase tracking quality, especially when occlusions occur.

The Tangent model proposed in [10] functions in a similar way where agents (as well as their personal area) are represented in the relative velocity space. However, here the goal is to compute the interaction area (where both agents are the closest) and, based on its position relative to the walkers' personal areas, a decision is made on which agent goes first and which gives way. The resulting adaptation efforts are share and asymmetrical as observed in real-life scenarios.

Finally Ondrej's Vision model aims to solve interactions based on information accessible from the visual flow, thus simulating a perception-reaction loop [7]. The agents rely on the bearing angle and the approximate time-to-collision to make their decisions. They are also capable of interacting with any kind of static or dynamic obstacle as long as they ï£¡seeï£¡ it.

These earliest velocity models for crowd simulation have served as a basis for many developments then. The remaining part of the class show the latest developments for the RVO-family of simulation algorithms. Before that, we wonder why/how this new class of simulation algorithms may improve the level of realism reached by simulations.

2.1.3 Experimental validation

The basic principle of velocity-based models is to compute the admissible velocity domain, i.e., the velocities at which an agent can move without provoking collisions in the near future. Each model proposes specific methods to compute this domain and to select a specific solution velocity. However, all of them are based on a linear extrapolation of the current situation to check for future collisions. Do humans perform such a prediction? Do they anticipate the future conditions of an interaction and do they react accordingly? To a certain extent, this hypothesis made by velocity-based models is validated by the following experiment.

We designed the experiment illustrated in Figure 2.1. The results of this experiment are completely detailed in [6]. We asked some participants to stand at the corners of a square area (25m. large) and asked them to walk to the opposite corner. We controlled their task with networked computers at each corner. We synchronized participants starting to provoke the situations of probable collision between two participants following orthogonal paths. We randomized passage orders and put occluding walls to prevent participants from reacting before they reached their comfort speed and seesaw each other. We recorded their trajectories using an optoelectronic motion capture system. We observed more than 400 pairwise interactions. The Eexperimental setup is illustrated in Fig. 2.1.

The variation of individual behavior, reaction delay and comfort speed actually changed the accurate conditions of the crossing. Then, we could analyze the correlation between initial conditions of

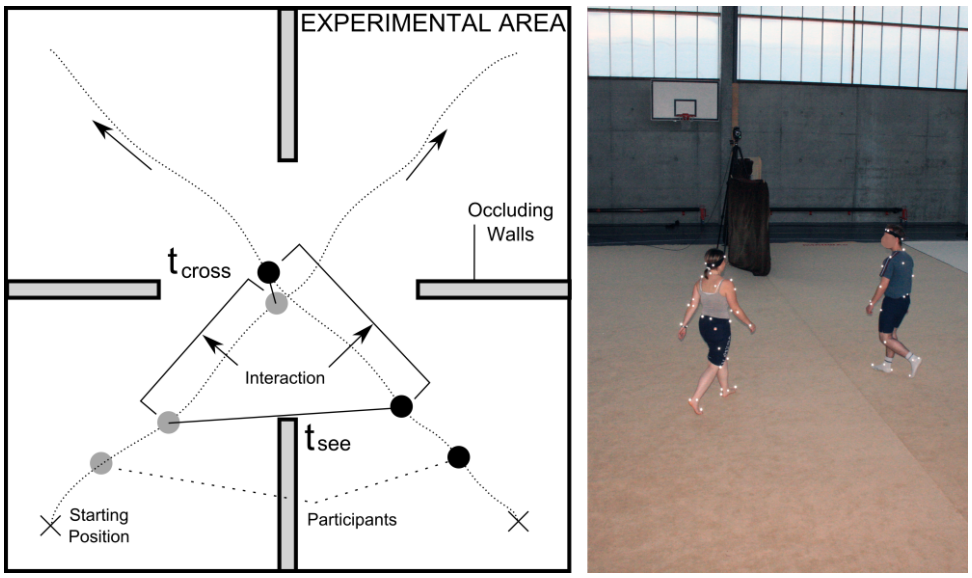


Figure 2.1: Left: illustration of the experimental setup to observe collision avoidance between two participants. The square experimental area is represented from top (25m wide). Right: picture taken during experiments. One can see the two participants who received a start signal following orthogonal trajectories and avoiding each other.

interactions with the existence and the nature of avoidance maneuvers. We defined MPD, the minimum distance at which participants would meet if they dido not perform any avoidance maneuvers, i.e., if they continued walking straight in their direction at constant speed. In other words, we compute an estimate of the closest approach by linearly extrapolating the participants' trajectories from current position and velocity. The MPD can be computed at any time: Fig. 2.2., left plot, displays an example of evolution of MPD in time, for the whole interaction phase (the interaction phase starts as soon as participants are able to see each other and ends when they pass at minimal distance). Time was normalized for each experiments and grewows from 0 (beginning of interaction) to 100 (end of interaction).

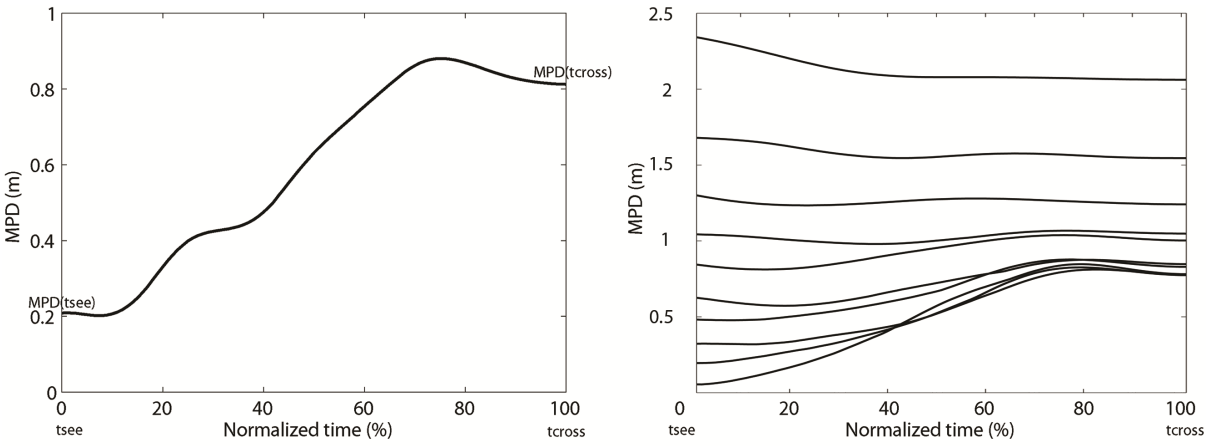


Figure 2.2: Left: MPD evolution in time during the whole interaction phase. Right: mean evolution of MPD during the whole interaction phase for groups of 40 experiments, ranked and grouped by order of initial MPD value.

In the example of Figure 2.2, left plot, one can observe a typical situation and its evolution in time.

At time t=0, MPD=0.2m, which means that if participants continue walking like this, a collision is predicted (distance is between body centers). Rapidly, MPD starts progressively growing up to 0.8m. This distance is no more critical and allows participants the avoiding collisions. The MPD value is regulated to this value during the last quarter of interaction. How can we interpret this evolution? We here show the clear sign of the ability of humans to predict the situation of future collisions and to react with anticipation. Indeed, by definition, the variation of MPD can only be explained by some maneuvers performed by walkers. We observe i) that MPD control by participants starts rapidly after they are able to see each other, ii) that the contribution of maneuvers is positive: MPD is monotonically increasing, iii) MPD is not exaggeratedly increased, iv) that maneuvers are over waylong before interaction is over (interaction duration is 4 seconds on average).

We described a specific example. We ordered our 400 experiments by the initial MPD value, and formed 10 groups of 40 experiments each. We computed the average evolution of MPD for each group. Results are reported in Fig. 2.2., right plot. We show that, when MPD is initially low (groups 1, 2, 3, 4, 5, and 6 with MPD < 1m at the beginning of the interaction phase), MPD is controlled by avoidance maneuvers and increased to an average of 0.8m. For other groups, no avoidance is observed and MPD remains constant on average (still, with some fluctuations, report to [6] for details).

The conclusions of our experiment are the following. First, humans are able to predict the future conditions of interactions with accuracy: avoidance maneuvers (i.e., variations of MPD) were observed only when required. Second, humans are able to react accordingly, in advance, to the benefit of the situation: the fact that avoidance maneuvers are over before people reach their closest approach proves anticipation. These two conclusions partially validate the foundations of velocity-based models and prove them to be closer to real human behavior in comparison with distance based models such as social forces models or other particle models.

2.1.4 3 examples of velocity-based models

The fundamental objective of velocity based models is to compute the admissible velocity domain, i.e., the set of walking velocities that prevent agents from enter into collision with static and moving obstacles in the near future, and to select a specific solution among this set. Various techniques were proposed to compute this domain. In this part of the class we describe, discuss and compare 3 solutions that were developed at Inria in the past 5 years. These models are denoted: the Paris model [8], the Tangent model [10] and the Vision-based model [7]. Related papers are provided in annex to these notes.

The Paris model

The Paris model proposes a discrete approach to estimate the admissible velocity domain. Each agent's motion is controlled by the direction of walking θ and its walking speed s. Let us consider an example of interaction between two agents, A and B. We describe below how the motion of A is controlled by the model with respect to the motion of neighbor agent B.

The future positions of B are predicted from the linear extrapolation of the current position and velocity vector (current time is time t_0). In Fig. 2.3, left image, the current position and velocity of A and B are shown, as well as the future positions of B at times t_1, t_2 and t_3.

Several time intervals are considered. For each time interval $[t_i, t_{i+1}]$, the angular sector covered by the predicted motion of B and relatively to the position of agent A is computed. In Fig. 2.3, the left image displays the angular sector covered by B between t_1 and t_2. The angular sector is delimited byθ_1 and θ_2. Agent A may enter into collision with B in the time interval $[t_i, t_{i+1}]$ if, and only if, A moves in a direction belonging to this angular sector.

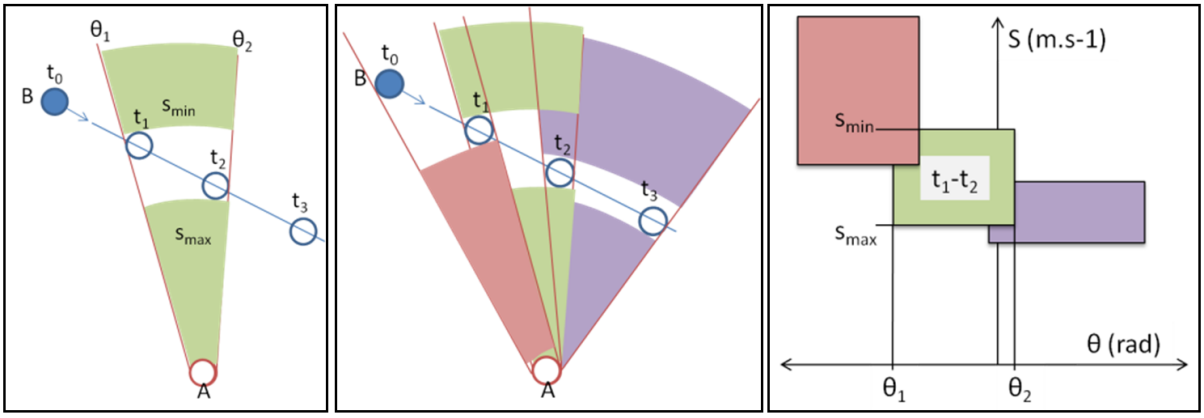


Figure 2.3: Illustration of the Paris model: Interaction between two agents A and B. Left: illustration of the time interval $[t_1 - t_2]$, of the angular sector $[\theta_1, \theta_2]$ and of the speed interval $[s_{min}, s_{max}]$. Middle: repetition of computations for successive time intervals. Right: representation of the deduced admissible velocity space in the control space.

For each time interval and corresponding angular sector, we additionally compute s_{min} and s_{max}, which are respectively the minimum speed at which A should move to pass in front of B, and the maximum speed at which A should move to give way to B. In the example of Fig. 2.3, if A moves in a direction $\theta \in [\theta_1, \theta_2]$ at a speed $s \in [s_{max}, s_{min}]$, there is a high risk of future collision. Fig. 2.3 illustrates these bounds (colored parts of angular sectors are safe speeds).

Steps 1-3 are repeated for each neighbor agent and for each time interval. Portions of inadmissible velocities (belonging to intervals $[s_{max}, s_{min}]$ and $[\theta_i, \theta_{i+1}]$) are successively reported into the control space (Fig. 2.3., right image: the space left blank corresponds to the admissible velocity domain). By construction, the admissible velocity domain is deduced. The model uses a cost function to deduce the best solution belonging to this domain (that minimizes deviations as well as distance to comfort speed).

As a conclusion, the Paris model iteratively computes the inadmissible velocity domain with some approximation. Indeed, s_{min} and s_{max} should be functions of time, but only the worst cases values are retained for one given time interval and one considered neighbor agent. The shorter the time intervals, the more accurate the model. But this approximation allows the model to be efficient in terms of computation time. In the original paper, to find a trade-off between accuracy and performance, Paris suggests to sample future time in an irregular manner: the first time intervals, which correspond to the imminent future, are shorter than the following ones (the paper suggests: $t_1 = 1s.$, $t_2 = 2s.$, $t_3 = 4s.$, $t_4 = 8s.$).

The model easily takes into account multiple interactions by looping steps 1-3 of the method for several agents. Each time, a new set of constraints is added to the control space. The model is also able to consider static obstacles: they are sampled and considered as sets of static agents. They are processed the same way as moving agents, at the exception of speed constraints. Only s_{max} is computed, s_{min} has no sense here.

The Tangent model

The Tangent model was designed to reproduce some experimentally observed interactions between two walkers with a very high level of realism, higher than with the Paris model. In particular, the Tangent model considers the human perception of othersï£¡ velocity, and introduces some error terms: their role is to delay avoidance maneuvers when accurate perception is not yet obtained. Doing so, the timing of interactions is reproduced. To our knowledge, the Tangent model is still the only one attempting

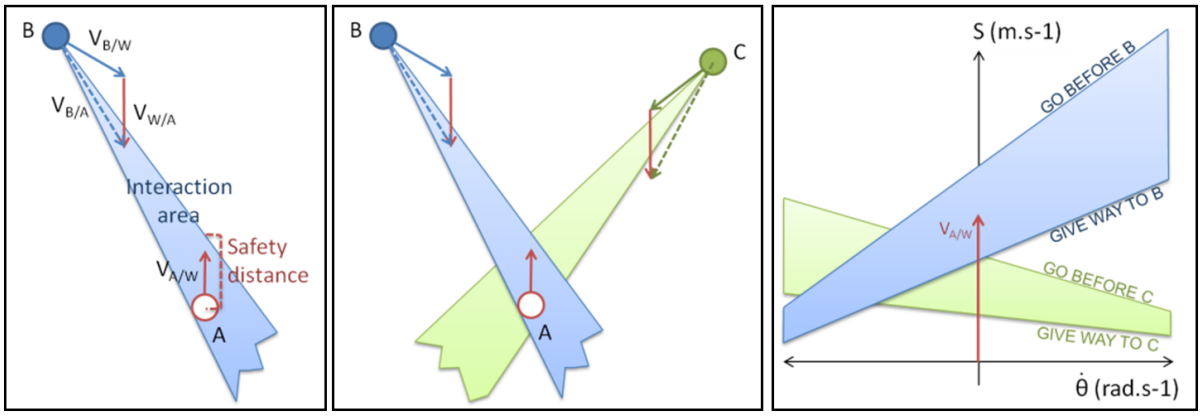


Figure 2.4: Principles of the Tangent model. Left: interaction between agents A and B. Middle: interactions between agent A and $(B + C)$. Right: illustration of the resulting constraint in the control space.

to correctly model interactions in time. The following paragraphs explain the basic principles of the Tangent model. A detailed description is to be found in [10].

The model is described from the example of motion control for agent A during interaction with agent B (see Fig. 2.4.). The velocity vector of B relatively to A, $V_{B/A}$, is first computed. By linear extrapolation, $V_{B/A}$ allows to estimate the distance at which B will pass A. When the crossing distance is too low, A has to perform an avoidance maneuver. To this end, we define a safety distance in front of A: we consider that the crossing distance is too low when $V_{B/A}$ belongs to the interaction area represented in Fig. 2.4., left image. To model when A should react, we consider in addition a perception error ϵ that decreases over time. A actually reacts when $V_{B/A} + \epsilon$ belongs to the interaction area.

To avoid a future collision, the relative velocity vector $V_{B/A}$ must lie out of the interaction area. Agent A can adapt this relative velocity by playing on its own velocity vector $V_{A/W}$ (we cannot assume that A controls Bï£¡s motion), we remind that:

$$V_{B/A} = V_{B/W} - V_{A/W}$$

where W refers to the World coordinate system. Fig. 2.4., illustrates these two components of the relative velocity vectors. In the example Fig. 2.4, one can see that A can for example decelerate: the $V_{A/W}$ component of $V_{B/A}$ would then be shorter and collision avoided. Equally, A could turn to the right.

These adaptations can be better appreciated form the control space point of view. Indeed, the interaction area is similar to two linear inequality constraints that can be projected onto the velocity space. Multiple interactions can then be solved as by solving a system of linear inequalities. Fig. 2.4., middle image, illustrates an interaction between agent A and two other agents $(B + C)$. In the right image, we project the constraints imposed by interaction in the velocity space. This representation enables an easy description of the situation and how to adapt the motion to avoid collisions. One can observe that the current velocity of A will provoke a future collision with B. A could decelerate to give way to B, but will then start interacting with C. In this specific example, one valid solution could be to turn to the right and slightly decelerate: A would give way to B but go before C.

The tangent model can consider static obstacles with geometries described by sets of line segments, as explained in [15]. In comparison with the Paris model, we can see some differences. The time dimension is taken into account: the model describes risks of future collisions, but does not explicitly

estimate when said collisions will occur. Instead, by introducing an error term, a reaction time is simulated. To solve interactions with respect to the imminence of collision risk, we actually order interaction with respect to time-to-collision. The system of linear equalities that captures multiple interactions is progressively built in this way until a limit number of interactions is reached or when the solution space becomes null. Doing so, agents react to interactions having a higher risk of collision.

The Vision Model

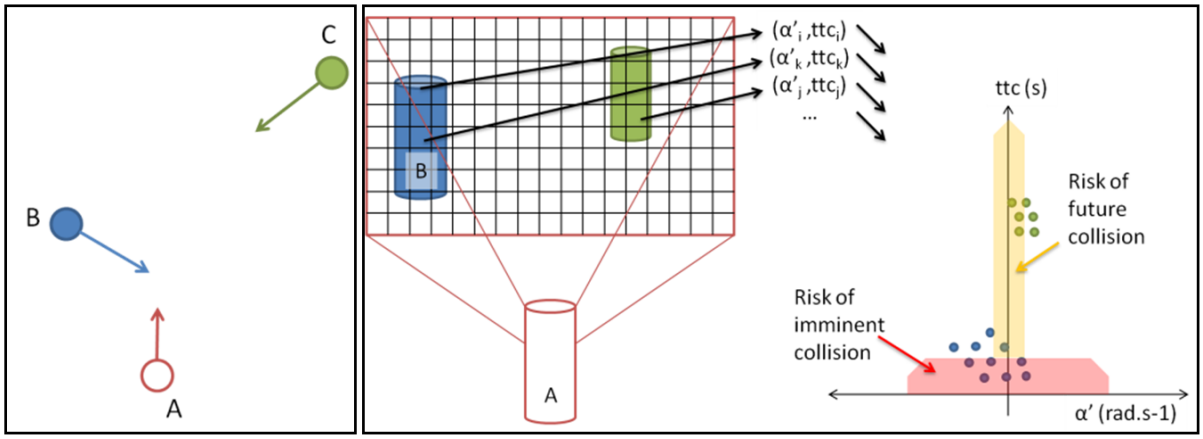


Figure 2.5: Principles of the Vision model. Left: situation of an interaction between agent A and two other agents $(B + C)$. Right: visual representation from the agent A's point of view is computed and projected into the $(\dot{\alpha}, ttc) - space$.

The two previous models assume that agents, i.e., simulated human walkers, are able to integrate a large quantity of information about neighborsï£¡ motions. This information is progressively projected into the control space to deduce the admissible velocity domain. Even though the Tangent model also models motion perception error to better simulate the timing of an interaction, real humans do not process information this way to control their locomotion. They control their walk mainly according to their visual perception of their environment. The objective of the Vision model, in comparison with the two previously proposed models, is to better simulate this perception-action loop.

The neuroscience field stated that humans, during avoidance of static or moving obstacles, successively answer two questions: will a collision with the obstacle occur? When will this collision occur? They react accordingly. The manner in which humans process their optical flow to answer these questions is still under debate but some theories state that two variables are directly exploited by humans for motion control: first, $\dot{\alpha}$, the derivative of the bearing angle, and second, ttc, the time-to-collision. When an obstacle is always visually perceived under the same angle (i.e., $\dot{\alpha} = 0$), and is growing in the image formed on the retina ($ttc > 0$), a risk of future collision is detected. The imminence of the collision risk is determined by ttc as well.

The Vision model reproduces this perception action loop. The principle of the model is illustrated in Fig. 2.5, from the example of an interaction between an agent A and two other agents $(B + C)$, as shown in the left image of the figure. To start, we compute a digital representation of the environment from the perspective of agent A (right image). This representation is computed similarly to classical graphical rendering techniques. A matrix represents the perceived image, each pixel is computed based on rastering techniques. The comparison stops here, we do not compute the pixels' graphical properties (color, intensity). Instead, for each pixel p_i, two values are computed: $(\dot{\alpha}_i, ttc_i)$, the time-derivative of the bearing angle and the time-to-collision. These values are deduced from the relative position and

velocity of the corresponding obstacle the pixel belongs to. It should be noted that an obstacle is possibly represented by several pixels (especially when close or big), and that each value may change for each pixel, because relative positions and velocities slightly change.

At the end of the rendering process, the notion of obstacles disappears, the agent A now interacts with a pixel cloud with various $(\dot{\alpha}_i, ttc_i)$ values. All this visual information is projected into the $(\dot{\alpha}, ttc) - space$. Agent motion control is performed according to a simple perception-action simulation loop:

1. Pixels with low $\dot{\alpha}_i$ values correspond to a risk of future collision. When such pixels are perceived, the agent turns to change this situation. The goal of the agent is taken into account when computing this anticipated reaction to avoid large deviations.

2. Pixels with low ttc_i values correspond to an imminent risk of collision (even with large $\dot{\alpha}_i$ values because of the body envelope). When pixels with such low values are perceived, the agent decelerates. He adapts his tangential speed to the lowest perceived ttc_i value.

In the example of Fig. 2.5., we explain how agent A's motion control is performed during an interaction with two other agents $(B + C)$. Agent B is perceived with low ttc pixels values whereas agent C is perceived with low $\dot{\alpha}_i$ pixels values. Both a risk of imminent and future collision. As a result, agent A will both decelerate and turn to solve this interaction. Pixel detection thresholds as well as motion control laws, which constitute the core of the vision model, are detailed in the original paper.

This model has interesting properties. It is able to consider any type of obstacle, static or moving and with any geometry, with undifferentiated processing because they are all reduced to a set of pixels. Also, the visibility of obstacles as well as their importance relatively to the place they occupy in the perception image is implicitly taken into account. Finally, the model was proven to be capable of simulating the emergence of well known pedestrian patterns under some traffic conditions.

2.1.5 Conclusion

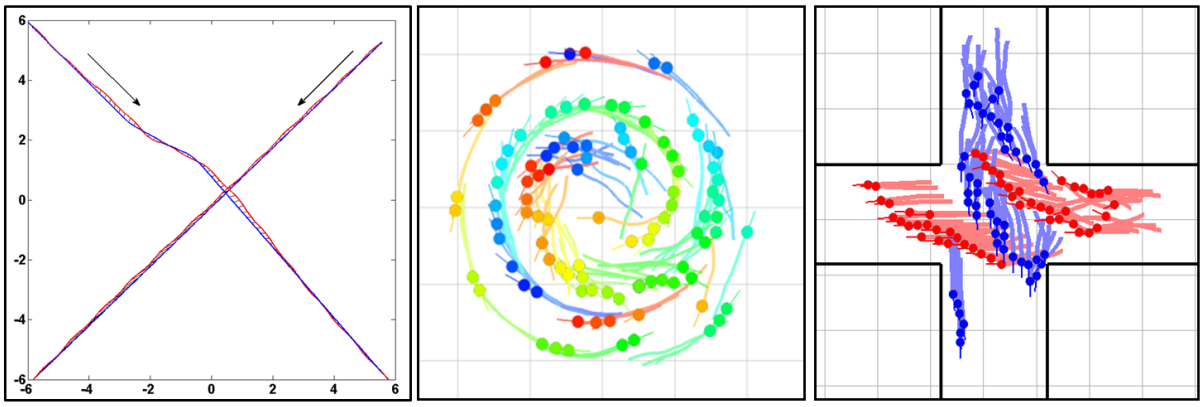


Figure 2.6: Simulation Results. Left: collision avoidance between 2 agents simulated by the Tangent model, output trajectories (in color) are superimposed with experimental data (black): situation can be accurately reproduced. Middle and Right: Simulation results for the Vision model. Emergent formation of pedestrians are observed, conform to real observations.

We successively described three different velocity-based models. They propose three different methods to compute the admissible velocity domain and to finally control the agentsï£¡ motion. They can be compared with respect to various criteria.

First, they were designed with different objectives in mind (see Fig. 2.6). The Paris model was, chronologically, the first velocity-based model: the objective was to enable anticipated adaptation for crowd simulation. But when a reaction should start? The tangent model answers this question by introducing a ï£¡motion perception errorï£¡ term in the formulation of the model. But both of these solutions remain far from the real perception-action loop that humans use to control their locomotion, which is modeled in the vision approach with yet unparalleled fidelity.

Second, they are based on different control spaces. The Paris model works in the orientation-speed space (we call speed the norm of the velocity vector). The Tangent model works in the angular-tangential-velocity space. Finally, the Vision model independently controls turning motions and speed (respectively from the existence of a risk of collision and its imminence). Next, all these models take into account obstacles with various geometries, which is a very important property and often neglected in other approaches. The Paris model samples obstacles as set of static agents. The Tangent model is able to simulate interactions between agents and line segments, which is very practical when considering building-like environments. Finally, the vision model considers indistinguishably any kind of obstacle (other agents or static obstacles) with any geometry, because motion control is performed from their graphical representation.

Also, they consider time-scale in various ways. The Paris model provides both risks of collisions and their situation in time: several time windows are successively explored. At the opposite, the Tangent model computes risks of collision, with one single TTC value (based on current velocities), but unlimitedly explores future time, as for the vision model.

Finally, both the Paris and the Tangent model need additional techniques to filter, order and select interactions (from the visibility of agents, their relative positions in space, the imminence of risk of collision, etc.). Indeed, one agent should not interact with all other agents if numerous (as real humans who only consider a neighborhood). The effect of the used notion of neighborhood on simulation was proven to be important in terms of emergent behaviors. The vision model is here more satisfying, as this selection process is implicitly based on obstacle visibility and relative importance (by their size in the perceived image). The vision model was proven to simulate the emergence of patterns of pedestrians with phenomena similar to reality. Is this property responsible for this interesting behavior?

Other questions still need addressing concerning velocity models. New types of interactions should be addressed. Recently, a new model for the following behavior was proposed, with a high level of realism as evaluated from experimental data. How to combine, for example, following and avoidance behaviors to simulate complex group behaviors, which is crucial in the aim of simulating natural looking crowds?

The fundamental basis of velocity models should also be put again into question. A simple linear extrapolation of trajectories based on current position and velocity is sometimes a bad prediction, especially during maneuvers (a small deviation may completely change the trajectory prediction between two time steps): in crowded places where people constantly adapt their motion, is using a velocity model useless? Probably not, because computations are constantly re-evaluated. But smarter predictions, at the level of real human abilities, would probably make this new type of model even more realistic and useful.

Bibliography

[1] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[2] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *Robotics Automation Magazine, IEEE*, 4(1):23–33, Mar 1997.

[3] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical Review E*, 51:4282, 1995.

[4] Anders Johansson, Dirk Helbing, and Pradyumn K Shukla. Specification of the social force pedestrian model by evolutionary adjustment to video tracking data. *Advances in Complex Systems*, 10(supp02):271–288, 2007.

[5] Rahul Narain, Abhinav Golas, Sean Curtis, and Ming Lin. Aggregate dynamics for dense crowd simulation. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, 2009.

[6] Anne-Hélène Olivier, Antoine Marin, Armel Crétual, and Julien Pettré. Minimal predicted distance: A common metric for collision avoidance during pairwise interactions between walkers. *Gait & Posture*, 36(3):399–404, 2012.

[7] Jan Ondřej, Julien Pettré, Anne-Hélène Olivier, and Stéphane Donikian. A synthetic-vision based steering approach for crowd simulation. *ACM Transactions on Graphics (TOG)*, 29(4):123, 2010.

[8] S. Paris, J. Pettré, and S. Donikian. Pedestrian reactive navigation for crowd simulation: a predictive approach. *Eurographics'07: Computer Graphics Forum*, 26((3)):665–674, 2007.

[9] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Computer Vision, 2009 IEEE 12th Int. Conf. on*, pages 261 –268, 29 2009-oct. 2 2009.

[10] Julien Pettré, Jan Ondřej, Anne-Hélène Olivier, Armel Cretual, and Stéphane Donikian. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proc. 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '09)*, pages 189–198, New York, NY, USA, 2009. ACM.

[11] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, New York, NY, USA, 1987. ACM.

[12] A. Schadschneider. Cellular automaton approach to pedestrian dynamicsŰtheory. In *In Pedestrian and Evacuation Dynamics*, pages 75–85, 2001.

[13] A. Treuille, S. Cooper, and Z. Popović. Continuum crowds. *ACM Transactions on Graphics (SIGGRAPH 2006)*, 25((3)), 2006.

[14] Jur van den Berg, Sachin Patil, Jason Sewall, Dinesh Manocha, and Ming Lin. Interactive navigation of individual agents in crowded environments. *Symposium on Interactive 3D Graphics and Games (I3D 2008)*, 2008.

[15] Yijiang Zhang, Julien Pettre, Jan Ondrej, Xueying Qin, Qunsheng Peng, and Stephane Donikian. Online inserting virtual characters into dynamic video scenes. *Computer Animation and Virtual Worlds*, 22(6):499–510, 2011.

2.2 Animation of crowd characters

EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

Tutorial
Introduction to crowd simulation

Julien Pettré
Inria

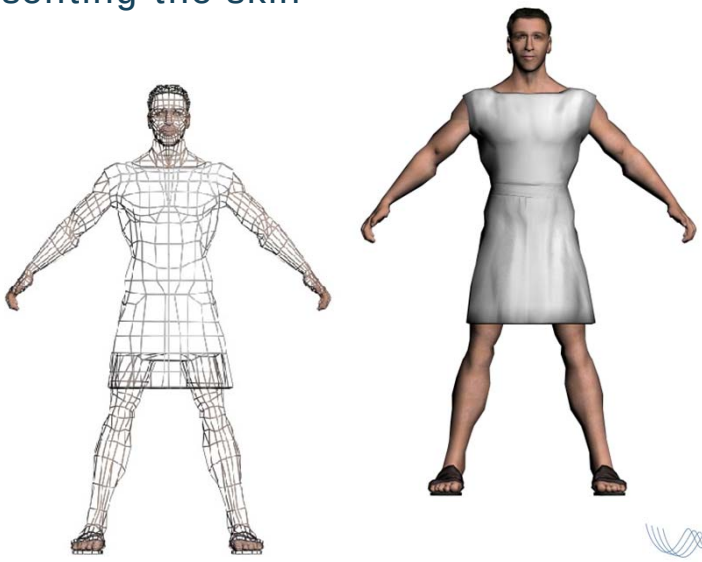
Nuria Pelechano
Universitat Politecnica de Catalunya

Character Animation

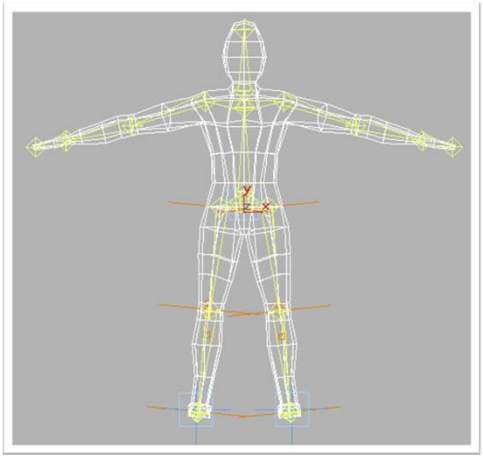
- Character Models
- Skin and cloth models
- Rigging
- Keyframing vs. Motion Capture

Character Models

- Triangle mesh representing the skin
  (+textures) and cloth



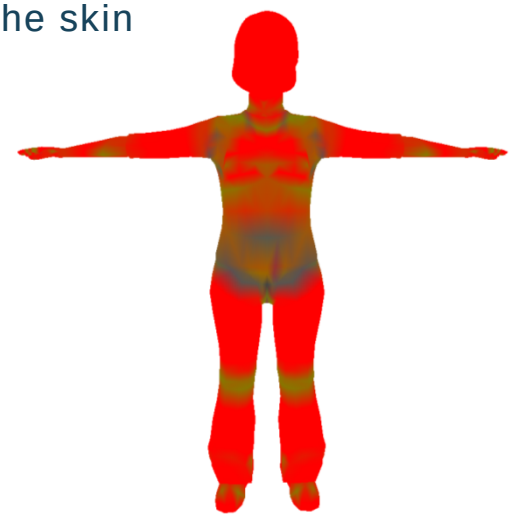
Character Models



- Triangle mesh representing the skin
  (+textures) and cloth
- Skeleton: hierarchy of bones

Character Models

- Triangle mesh representing the skin
  (+textures) and cloth
- Skeleton: hierarchy of bones
- Rigging
- Skinning



Palette Skinning

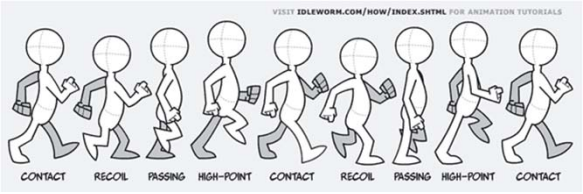
- Crowd animations: usually skeletal based
  - Linear blend skinning
  - Dual quaternions
- Matrices stored in GPU

		Key 0	Key 1	...
Bone 1	Matrix Row 0			
	Matrix Row 1			
	Matrix Row 2			
Bone 2	Matrix Row 0			
	Matrix Row 1			
	Matrix Row 2			
	↓			

EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

Character animation

- Walk Cycle: A looping series of positions
  - Shifting the animation provides the appearance of walking



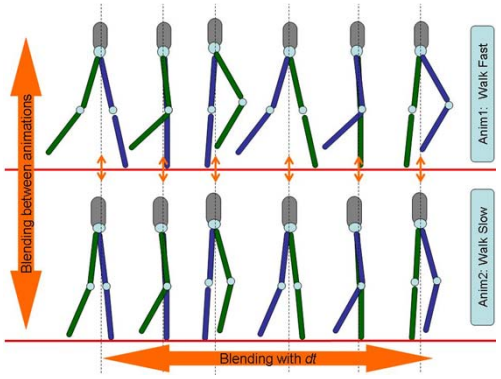
  - Animations can be exported with or without root displacement

How we can create new animation sequences from existing data?

EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

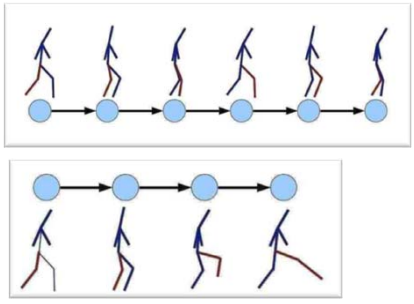
Blending and Time Warping

- Interpolating between animation clips:
  - Blending weights
  - Timing (fadeIn/fadeOut)
  - Time alignment
- Time Warping:
  - Interpolation within animation clips



Creating animations with Motion Graphs

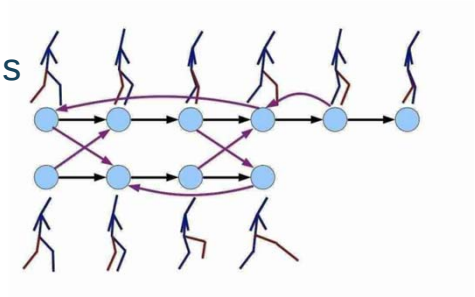
- Every motion clip is a graph
- Vertex = pose
- Edge = transition frames
- Each captured animation is a graph



Motion Graphs. L. Kovar, M. Gleicher F. Pighin, SIGGRAPH 2002.

Creating animations with Motion Graphs

- Every motion clip is a graph
- Vertex = pose
- Edge = transition frames
- Each captured animation is a graph
- Add transitions between similar poses



Motion Graphs. L. Kovar, M. Gleicher F. Pighin, SIGGRAPH 2002.

Available tools and SW

- Mixamo https://www.mixamo.com/
- Autodesk® Character Generator https://charactergenerator.autodesk.com/
- Cal3D http://home.gna.org/cal3d/
- HALCA http://www.lsi.upc.edu/~bspanlang/animation/avatarslib/doc

EUROGRAPHICS2017 The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

Games Engines

- Unity:
https://unity3d.com/

- Unreal Engine:
https://www.unrealengine.com/

How do I get my character to walk through my VE?

EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

Animating characters

- Footstep based trajectories
  - Following a given footstep trajectory
    - e.g: obtained from inverted pendulum model
  - Planning at the footstep level
- Root based trajectories:
  - Driven by velocity vector
    - e.g: joystick input
  - Driven by velocity vector and position of COM
    - e.g: typical output of a crowd simulation system
  - Driven by velocity & orientation

Footstep based trajectories

EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

Footstep based trajectories

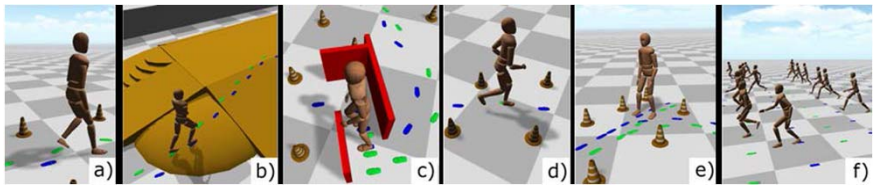
Following a given footstep trajectory

Synthesizing Motion Following Footsteps

- Goal
  - Online animation synthesis for footsteps simulators

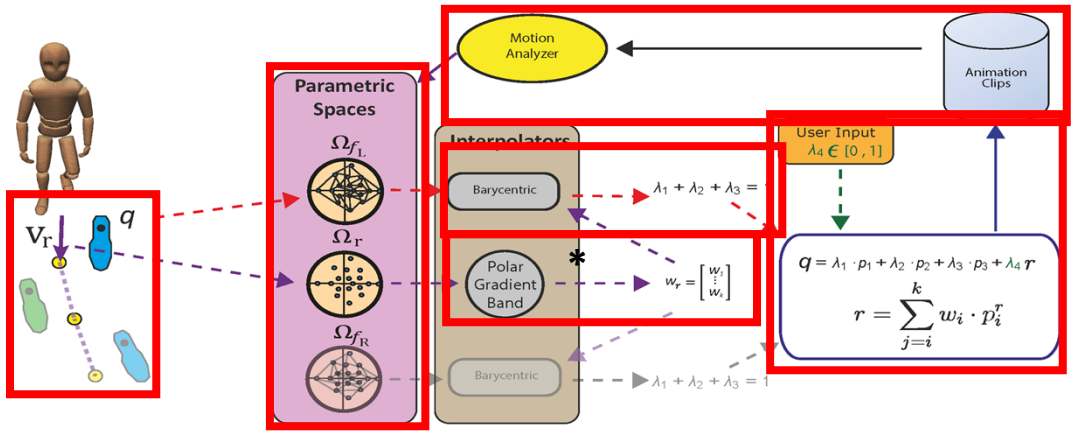
Singh S., Kapadia M., Reinman G. and Faloutsos P. Footstep navigation for dynamic crowds. *Computer Animation and VirtualWorlds*, volume 22(2-3):pp. 151–158 (2011).

  - Satisfy foot placement constraints
  - User control over the trade-off between footstep accuracy and root velocity

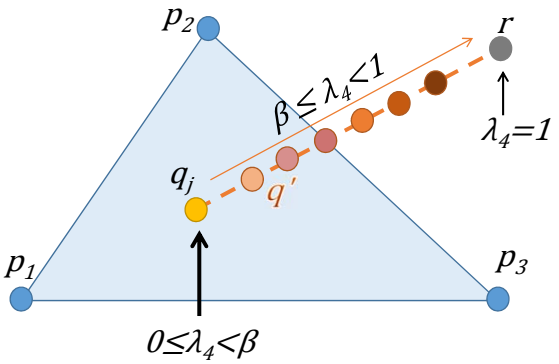


EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

4.2. Synthesizing Motion Following Footsteps



* Johansen R. Automated Semi-Procedural Animation. *Master Thesis* (2009).



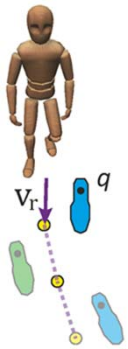
$$r = \sum_{i=1}^{k} w_i \cdot p_i^r$$

$$q_j = \lambda_1 \cdot p_1 + \lambda_2 \cdot p_2 + \lambda_3 \cdot p_3 + \lambda_4 \cdot r$$

$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 \cdot \sum_{i=1}^{k} w_i = 1$$

One step at a time: animating virtual characters based on foot placement

Egges, Arjan, and Ben van Basten. *The Visual Computer* 26.6-8 (2010): 497-503.



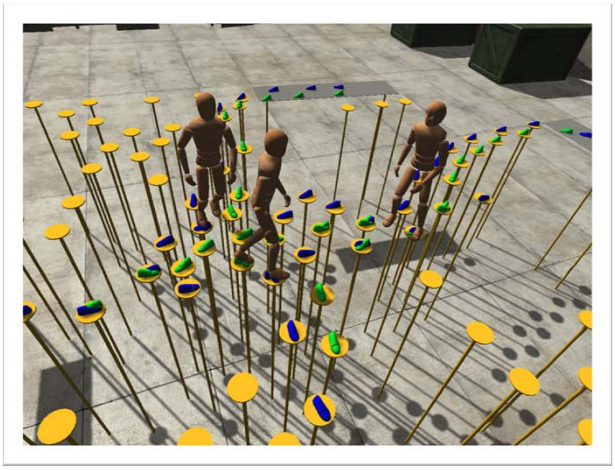
EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

Footstep based trajectories

Planning at the footstep level

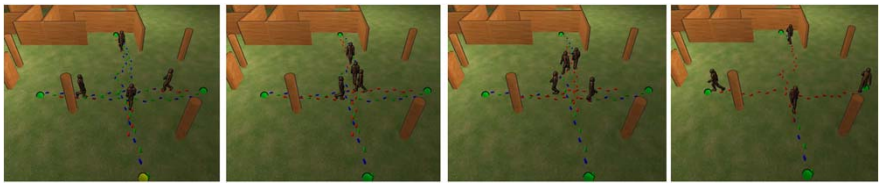
EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

Footstep based trajectories



Planning using Footsteps

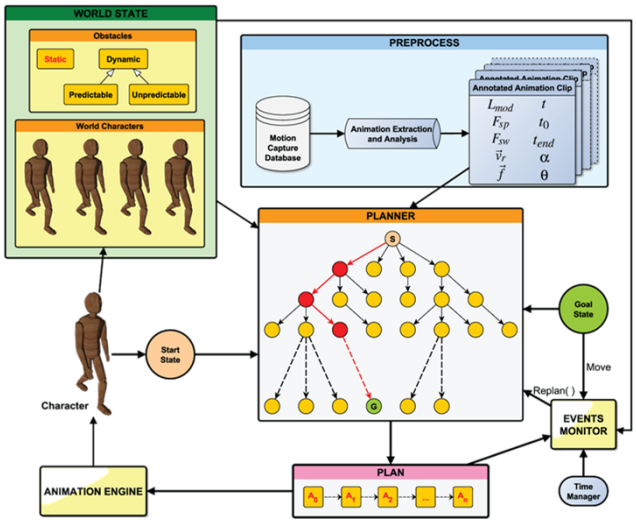
- Goal
  - Computation of natural footsteps trajectories for groups of agents



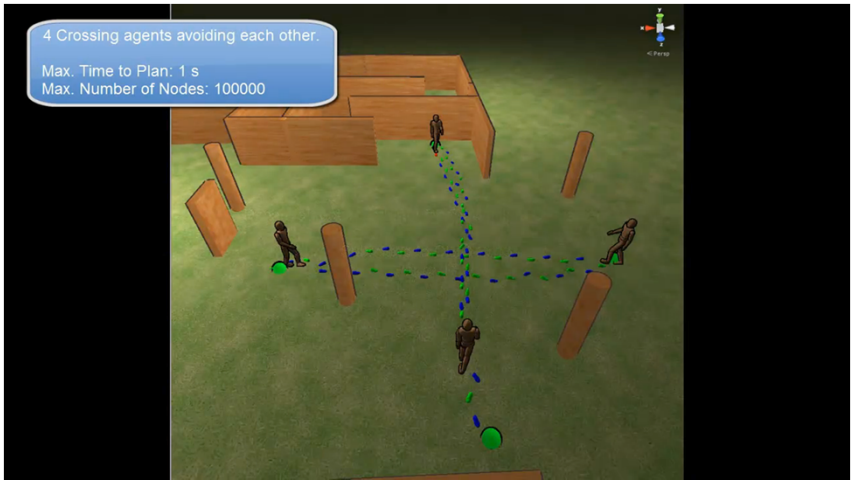
  - Accurate spatio-temporal foot placement
  - Fast computation: from a small set of animation clips outputs a sequence of footsteps
  - Dynamic method

EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

Planning using Footsteps

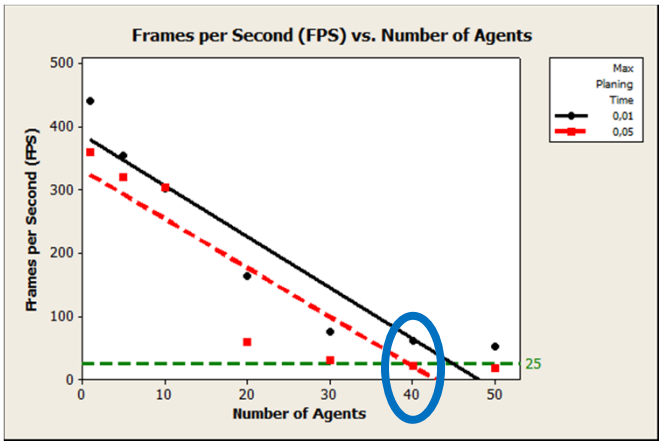


Planning using Footsteps



Planning using Footsteps

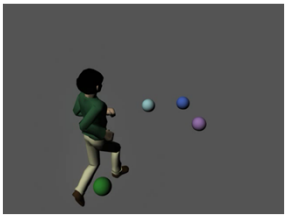
- Results
  - Unity Game Engine
  - Over 40 agents in real-time (Intel Core i7-2600k @ 3.40GHz 16 GB RAM)
  - 28 motion captured animations
  - FPS results depend on planning time



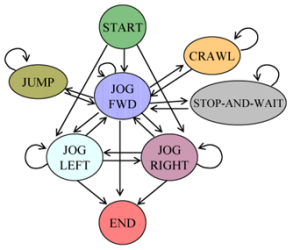
Planning using Footsteps

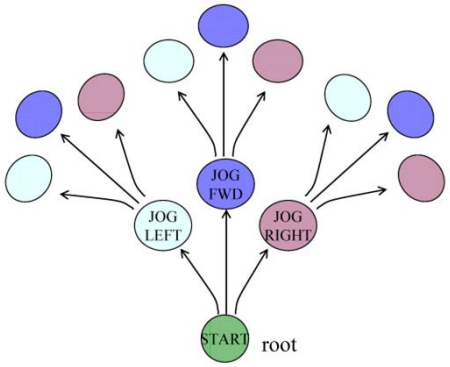
- Pre-computed search trees

Lau, Manfred, and James J. Kuffner. "Precomputed search trees: planning for interactive goal-driven animation." *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 2006.

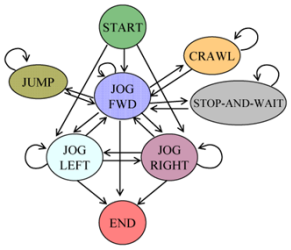


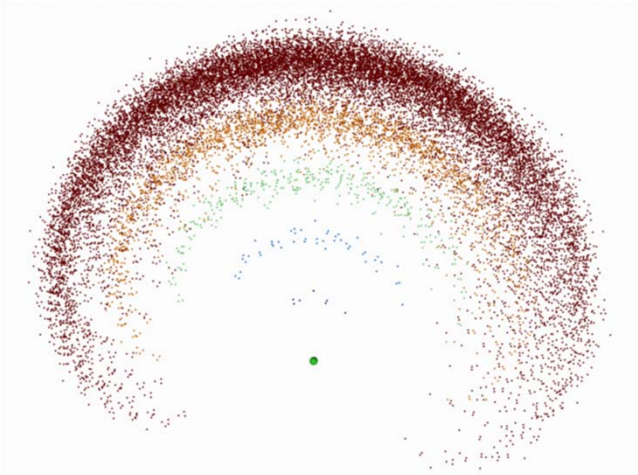
- Pre-computed search trees





- Pre-computed search trees





EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

Root based trajectories

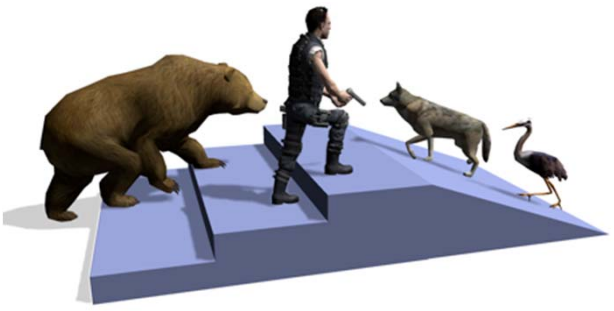
Root based trajectories

Driven by velocity vector

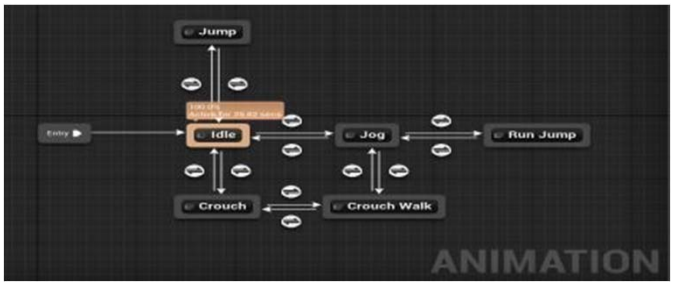
Animation driven by velocity vector

- Typical approach in video games to drive the animations of the character representing the player

R. S. Johansen, Automated Semi-Procedural Animation, Master Thesis. URL http://runevision.com/thesis/



- Unity: Mecanim
  - Blend trees: allows to create animation by blending between two pre-existing animations (e.g. to create an animation half way between walking and running)
  - IK to make small adjustment (e.g. foot positioning)



EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

- Unreal
  - Skeletal animation
  - Animation Blueprints: graphs to perform animation blending, control bones of a Skeleton, or setup logic that will define the final animation pose
  - Blend Spaces: allow for blending of animations based on the values of two inputs

Root based trajectories

Driven by velocity vector and position of COM

EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

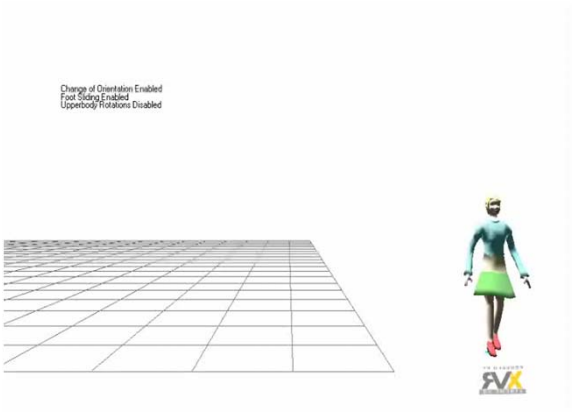
Root based trajectories: COM+velocity vector

- Most typical output from crowd simulation systems
- Problem:
  - To map the COM trajectory into a smooth animation
  - To interpolate correctly between animations
  - To avoid artefacts such as foot sliding
  - Velocity vector vs. orientation vector: (holonomic/non-holonomic movement)
- Any inconsistency between the output of the crowd simulation and the animation will produce artefacts in the simulation

Avatar Locomotion in Crowd Simulation

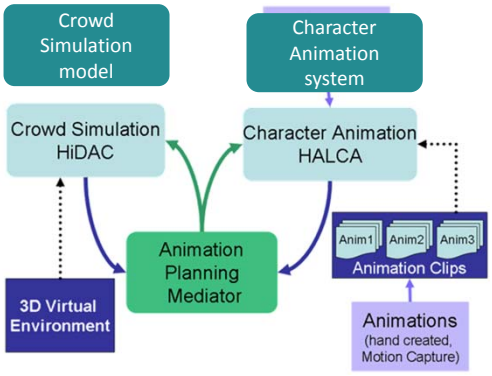
Pelechano, N., Spanlang, B. Beacco, A. *International Journal of Virtual Reality* 10.1 (2011): 13. CASA 2011

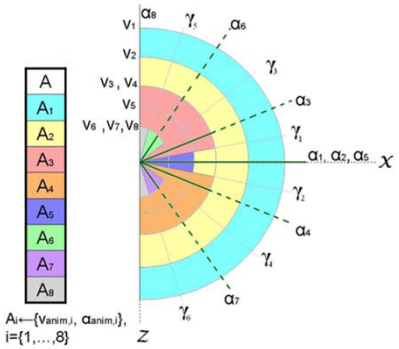
- Mapping velocity & orientation into animation controller:
  - Map velocity to steer the character
  - Map orientation to get the character facing the forward direction
  - Need to adjust spine rotation



EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

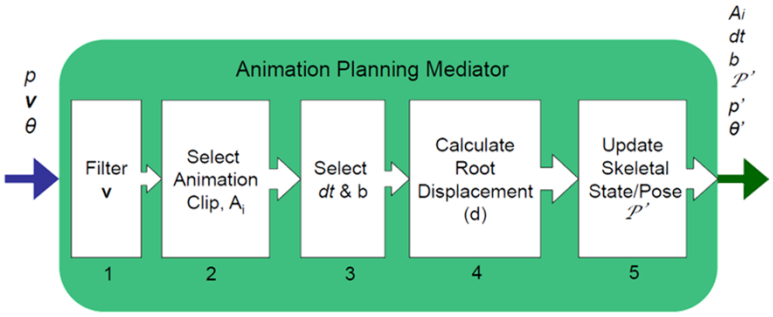
Reflecting the Root Motion



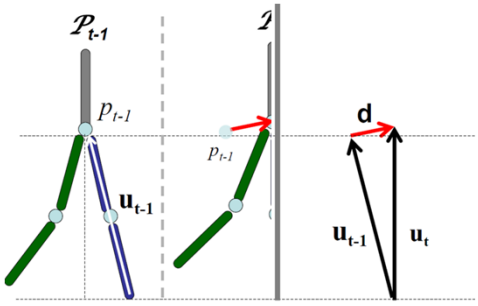


Reflecting the Root Motion

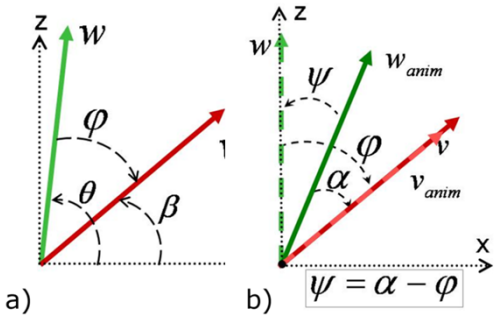
- Pipeline



Reflecting the Root Motion



Avoiding foot sliding



Correcting Torso Orientation

EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

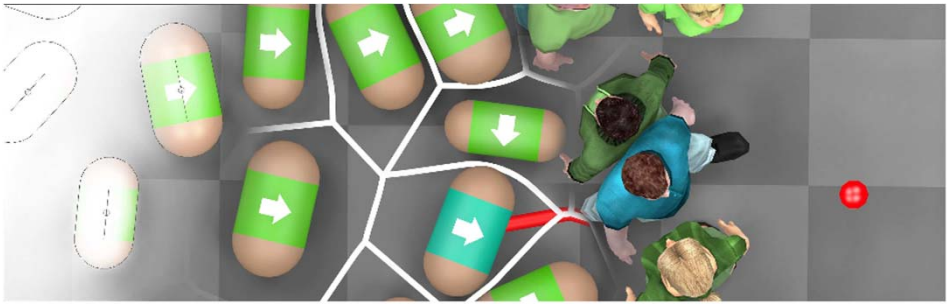
Root based trajectories

Considering velocity and orientation

EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

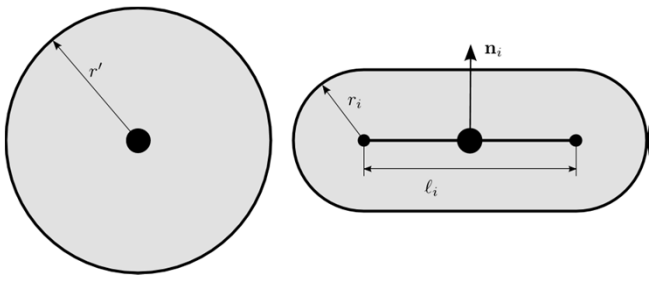
Torso Crowds

S. Stuvel; N. Magnenat-Thalmann; D. Thalmann; A. F. van der Stappen; A. Egges, in *IEEE Transactions on Visualization and Computer Graphics* . 2016



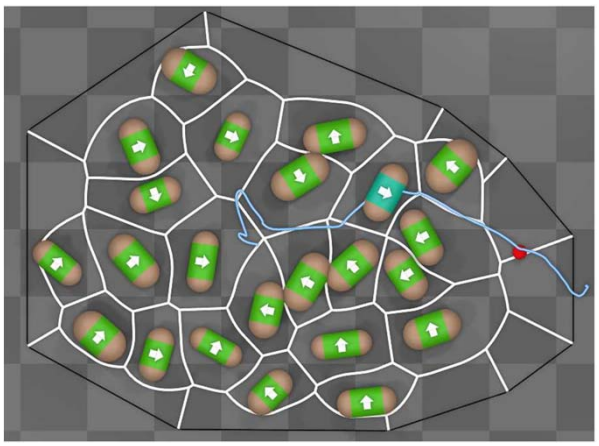
From left to right: the agent representation, calculation of the Voronoi diagram, planning a path towards a goal position, and finally the animation of virtual characters.

Torso Crowds



Torso Crowds

- Capture
- Simulate with Voronoi diagrams



Torso Crowds

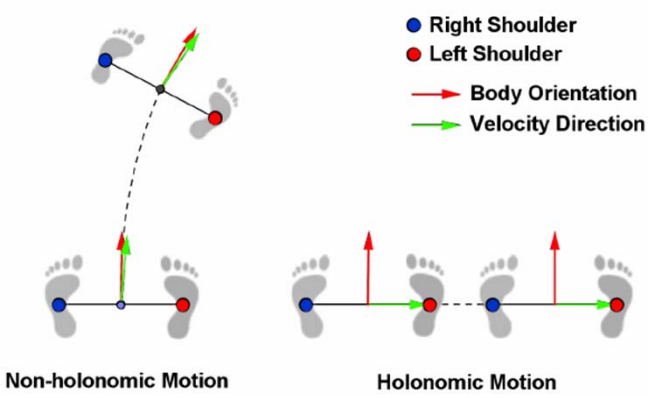
- plan the movement of an active agent, the following steps are taken:
  1. Find paths by exploring the vicinity in the GVD of the Voronoi cell containing the agent.
  2. Compute a score for each path, and determine the best-scoring path.
  3. Compute the desired agent orientation at the start of the path, accounting for available clearance.
- Differentiates 2 types of agens:
  - Passive
  - Active

EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

Holonomic collision avoidance for virtual crowds

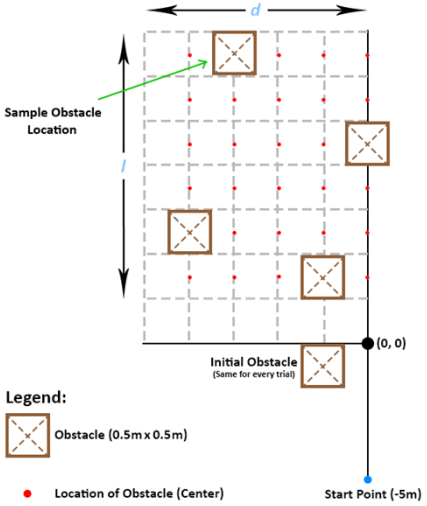
- Planning movement considering both velocity and torso orientation?

R. Hughes, J. Ondřej, J. Dingliana. 2015. In *Symposium on Computer Animation* (SCA '14). 103-111.

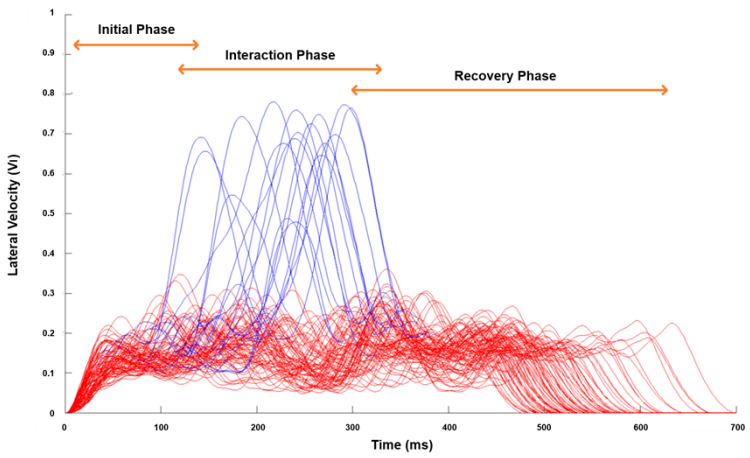


Holonomic collision avoidance for virtual crowds

- Capture such movement:



Holonomic collision avoidance for virtual crowds

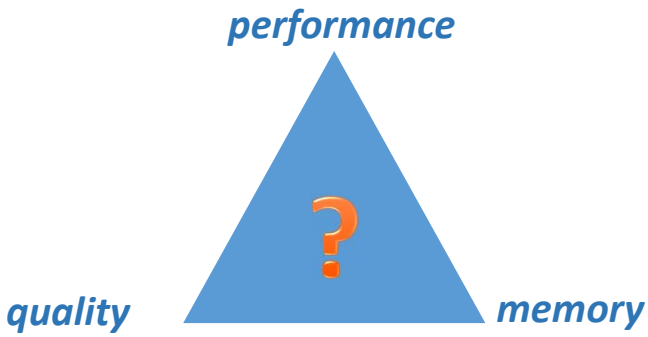


EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

Rendering Crowds

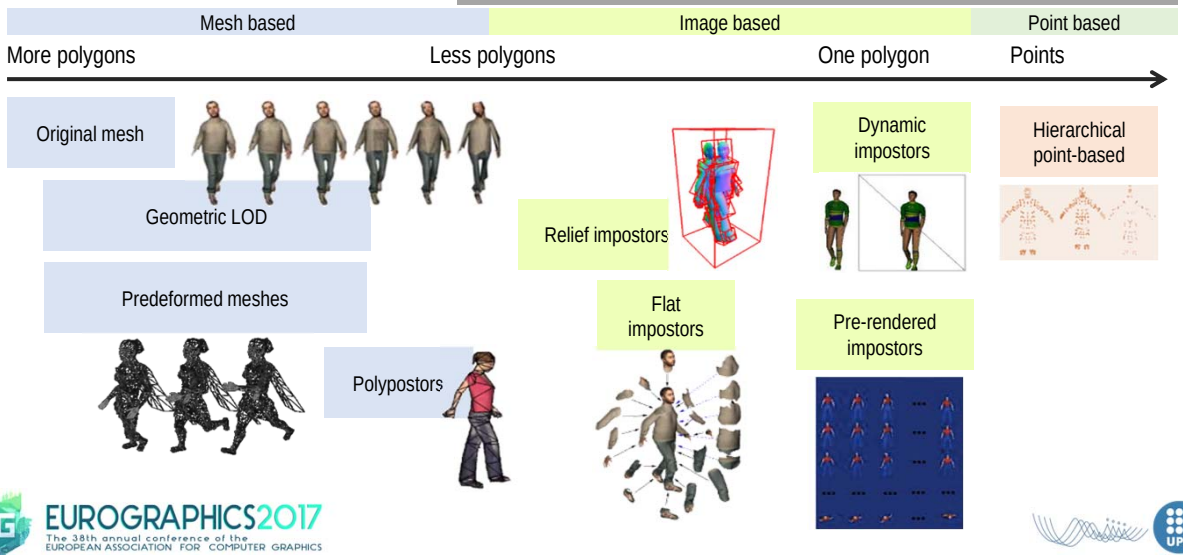
EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

Rendering Trade-off



Overview

A Survey of Real-Time Crowd Rendering A. Beacco, N. Pelechano, C. Andujar. *Eurographics - State or The Art Reports*. 2016.



EUROGRAPHICS2017
The 38th annual conference of the EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

HW improvements

- Instancing
- Move computation to **GPU** (skinning, animation blending, and even simulation)
- Tesselation shaders to add
- Limiting factors:
  - GPU memory
  - CPU-GPU bandwidth

Part 3

Annexes

EUROGRAPHICS 2007 / D. Cohen-Or and P. Slavík
(Guest Editors)

Volume 26 (2007), Number 3

Pedestrian Reactive Navigation for Crowd Simulation: a Predictive Approach

Sébastien Paris Julien Pettré Stéphane Donikian

IRISA, Campus de Beaulieu, F-35042 Rennes, FRANCE
{sebastien-guillaume.paris, julien.pettre, donikian}@irisa.fr

Abstract

This paper addresses the problem of virtual pedestrian autonomous navigation for crowd simulation. It describes a method for solving interactions between pedestrians and avoiding inter-collisions. Our approach is agent-based and predictive: each agent perceives surrounding agents and extrapolates their trajectory in order to react to potential collisions. We aim at obtaining realistic results, thus the proposed model is calibrated from experimental motion capture data. Our method is shown to be valid and solves major drawbacks compared to previous approaches such as oscillations due to a lack of anticipation. We first describe the mathematical representation used in our model, we then detail its implementation, and finally, its calibration and validation from real data.

1. Introduction

This paper addresses the problem of virtual pedestrian autonomous navigation for crowd simulation. One crucial aspect of this problem is to solve interactions between pedestrians during locomotion, which means avoiding inter-collisions. Simulating interactions between pedestrians is a difficult problem because its complexity grows rapidly with respect to population density. Also, obtaining realistic results is challenging: humans are used to observe navigating pedestrians in the real life and immediately detect artifacts in simulations. We present a reactive navigation technique for application in the domains of architecture, security, space ergonomy, and also the entertainment industry. We expect natural crowd motion emerging from a realistic microscopic pedestrian simulation.

Our solution for solving interactions between pedestrians is predictive and agent-based. Inputs are the definition of an environment, the current state and the destination of each pedestrian - destination is a desired direction derived from a navigation plan. The method first checks on future interactions between pedestrians: the evolution of pedestrians' position is predicted from an extrapolation of their current state. When needed, a long term avoidance motion is computed by taking into account these predictions. Our resulting microscopic pedestrian simulation model is calibrated and validated using motion capture data. Data are acquired according to two successive protocols. First, we measure interactions between two participants and use the resulting data to calibrate our model. In a second stage, we push the number of participants to the limits of our motion capture system abilities and let them navigate among obstacles, allowing us to compare the measured data with our simulation results.

Our first contribution is to solve major drawbacks in previous microscopic approaches, such as oscillations and jams. We believe these drawbacks were due, firstly, to the lack of anticipation, and secondly, to the simplicity of the reaction computation technique from the observed situation. Our second contribution is to propose a motion capture-based calibration of the model and validation of our results. Validation is generally done using hand-processed video sequences. Motion capture data are more accurate than those derived from video sequences: this allows us to decompose precisely in time and space how humans react to potential collision with others. Particularly, we could extract a criterion to detect the need for a reaction and compute adequate corrections to the trajectory.

Section 2 states our contributions with comparison to previous approaches. Section 3 describes our method to solve interactions between pedestrians from a technical point of view: firstly, how it is integrated into a crowd simulator, sec-

ondly, how pedestrians perceive their surrounding environment, and thirdly, how a reactive motion is computed. Section 4 describes the protocols and the results of the experiments used to calibrate the model parameters and to validate the approach. We then present some simulation results in usual architectural configurations. Finally, we conclude and provide perspectives to our work.

2. Related Work

One of the most important skills of a human being is her ability to navigate inside her environment. Even if this navigation task is one of the most basic behavior in real life, it is not yet solved correctly in a virtual world. First to allow people to navigate, they should be able to perceive their environment but not only in a geometric way. Studies in psychology and urbanism have shown that visibility and topology are also important in the navigation task. A structured and informed environment has to be used for path planning and reactive navigation of virtual humans in real time. The simplest task, for a pedestrian walking in a street, consists in minimizing possible interactions, which mean avoiding static and dynamic obstacles. Goffman [Gof71] describes techniques used by pedestrians to avoid bumping into each other. The social link between strangers is characterized by silence and indifference and to perform that, different behaviors are used. The first technique called externalization concerns the way that people are constantly making others aware of their intentions in order to minimize the interaction. Pedestrians selectively gather externalized information from other people by a second technique called scanning. The third technique, called the minimization of adjustment, expresses that people adjust their trajectory several meters before the conflict to make it perceptible early by others with the objective to reduce interaction and avoid coordination. Goffman introduces the notion of an oval security region whose front distance corresponds to an anticipation area depending on the pedestrian speed, while the width is the accepted gap to pass beside a person or an obstacle or to follow a wall. He also defines the law of minimal change meaning that a pedestrian will try in its journey to reduce the amount and the amplitude of turns. These studies illustrate the importance of prediction and anticipation in the navigation task.

It is known that in crowd motions, pedestrian flows walking in opposite direction generate their splitting to create dynamically some bands of pedestrians walking in the same direction. When the density of pedestrians becomes very high, it is possible to approximate the overall behavior of the crowd by using the laws of fluid evolution [TCP06]. In panic situations, pedestrians wish to move more quickly than usual and, forgetting all social rules, accept to be in a physical contact with their neighbors. Due to this physical interaction situation, they are developing a mimetic behavior consisting in reproducing the behavior of preceding characters in the flow. For Yamori [Yam98], this is obligatory followed by the notion of regulation, learned as a normative element by people living together inside the same macro structure or institution. However, as emphasized by Musse et al. [MT01], some small groups can be the motor and modify the behavior of biggest units such as a crowd, playing the role of the core group. Boles [Bol81] has observed the existence of a band structure inside a crowd of pedestrians moving on a sidewalk and explain this by its optimal configuration to regulate opposite flows. One of the most crucial problems to be solved, as pointed by Yamori, concerns the relation between microscopic and macroscopic structures and behaviors inside the crowd. The goal is to explain how an individual entity is constrained by the institution and on the other hand how the community impacts on the individual behavior along time. Yamori focuses his research in the formation of macroscopic band structures and postulate that such kind of macroscopic structure requires a critical density of population to emerge from the set of individual behaviors.

Macroscopic simulation has been historically the first approach to be studied to simulate the pedestrian displacement, due to its low calculation cost. In this approach the pedestrian is not treated on its own but as a component of a more macroscopic element [Hen71, PM78]. These macroscopic models are often used for animation purpose, like by Sung et al. [SGC04], to provide a globally convincing crowd motion. Another approach called microscopic simulation consists in handling the individual navigation of all moving entities. In that case, a system allowing dynamic collision avoidance is necessary to achieve consistency and realism. Several approaches can be distinguished such as particle and flocking systems. Particle systems are based on physical laws describing attractive and repulsive forces that can be associated to obstacles and moving entities. Forces applied to an entity are summed to calculate its new motion direction and speed [HFV00, BMdOB03, LKF05]. This model assimilates the displacement of an entity in the case of a high density to the motion of a particle inside a restricted area. I. Peschl [Pes71] justifies the use of this model in the case of an emergency situation with a high density of population. Particle based models allow the generation of a macroscopically plausible behavior in case of a high density, but they do not take into account anticipation, perception, or social rules. Moreover, close inspection of individual trajectories show some oscillations and unrealistic behaviors such as backward motion of the last people repulsed by the preceding one in a queue and many change of orientation along the path due to the interaction with other moving entities and static obstacles. Another drawback of this approach is its requirement of a small time-step for convergence purpose. Flocks are rule-based systems defining the behavior of an entity according to the behavior of the nearest entities [Rey00, BLA02]. It is well adapted for the collective motion of a group of animals following a leader but less for the variety of behaviors that can be observed in a sparsely populated crowd of humans.

Loscos et al. [LMM03] use a fine regular grid to handle reactive navigation and to store information about pedestrian movements enabling the emergence of flows of pedestrians. In the same way, Shao et al. [ST05] use a quadtree map for the path planning and a fine regular grid for obstacle avoidance.

Lakoba et al. [LKF05] argue that two points are necessary to improve the existing models: add decision-making capabilities and compare simulation results against measured data on pedestrian dynamics. Instead of classical models of crowd simulation based on fluid dynamics or particle systems which are only valid in very dense crowds, S. Goldenstein et al. [GKM*01] have proposed a multi-layer approach to model the behavior of crowd participants. We are also working on a multilevel model of each human allowing us to simultaneously take into account attraction/repulsion mechanisms such as in particle systems, dynamic computation of the neighborhood for sparse crowds, the management of social rules, path planning and activity planning. In this paper, we are focusing on the reactive navigation model and on the use of experimental data to validate the approach and calibrate the model.

3. Prediction and Resolution of Interactions

3.1. Principle

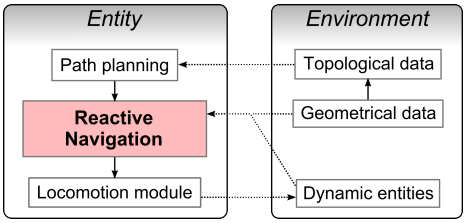


Figure 1: *The simulation architecture.*

The problem of reactive pedestrian navigation is part of the crowd simulation problem, and our method is included in a global architecture as shown in Figure 1. More details on the environment management and on dedicated path planning techniques we developed can be found in the literature [PDB05,PDB06]. The reactive navigation role is to steer entities in a realistic manner with respect to two possibly conflicting inputs: the goal of the considered pedestrian resulting from the path planning stage, and the current state of the environment, especially the presence of other pedestrians. The output we search for are updated speed and orientation allowing the pedestrian to avoid any static or moving obstacle while satisfying constraints of realism.

Our approach to this problem is a predictive one. For each entity, at desired rates, we search for a solution-move satisfying constraints and guaranteed to remain valid for a desired time window (at least the period at which reactive navigation is invoked). The key-idea is to model the environment

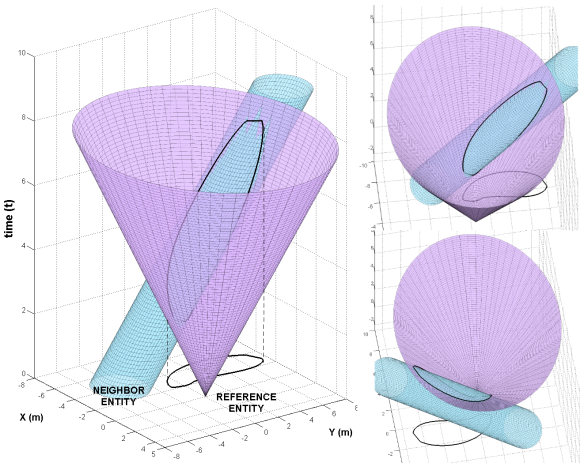


Figure 2: *Modeling the interaction between a reference entity and a neighbor entity in the (x,y,t)-space. The predicted trajectory for the neighbor entity is the blue cylinder, whereas the reachable space for the reference entity is the violet cone. Their intersection delimited in black is a future collision area (also projected in the plane).*

as shown in Figure 2 in the (x,y,t)-space, with (x,y) the horizontal plane and t the simulation time. We explore the reachable space of the *reference entity* in any direction and for a range of speed values, and search for possible collisions with *neighboring entities*. Figure 2 illustrates such an exploration for a given reference entity speed value. As any move direction is envisioned, the reachable space is then represented in the (x,y,t)-space as a cone whose opening angle depends on the considered speed value. *Neighboring entities* are then taken into account: they are represented as circles – whose radii are the sum of both the reference and the considered entity radii – moving along a predictive trajectory, computed from the current position, speed and orientation of the neighboring entity. Thus, the neighboring entity is modeled in the (x,y,t)-space as an elliptic cylinder. Consequently, the intersection of the cone and the cylinder delimits a *collision area* that the reference entity shall avoid. The difficulty of the problem is brought to mind looking at Figure 3, where different shapes of the collision area are displayed according to various solution speed values. Additionally, the figure does not represent the possible presence of several neighboring entities and of static obstacles which obviously increase the problem complexity drastically. As a result, we choose to base our solution on a discrete-time expression of the same modeling in order to avoid the problem complexity.

Three main steps, detailed in the next sections, allow us to compute the best speed and orientation for the reference entity:

1. Neighboring dynamic entities are first taken into account. From this, we deduce some sets of speed and orientation

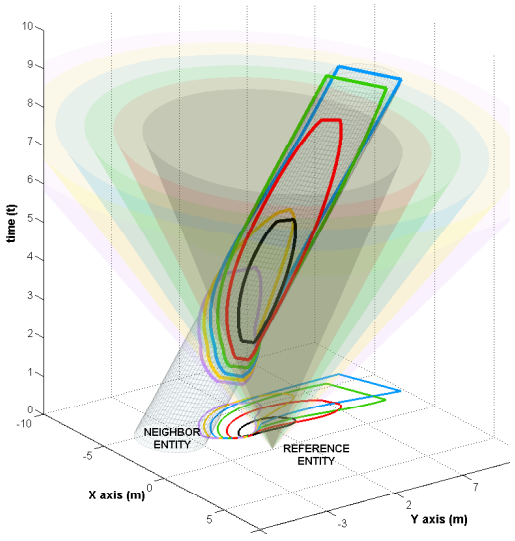


Figure 3: *We explore the reachable space for the reference entity for a given range of speed. The opening angle of the cone changes accordingly in the (x,y,t)-space, as well as the intersection with the predicted trajectory of the neighbor entity (each color corresponds to a different cone opening angle, and thus to a different reference speed value).*

ranges that allow collision free motion for a future time window.
2. In the same manner, static obstacles are considered. We deduce new valid sets of speed and orientation ranges.
3. Previous valid solution ranges are merged, scored and compared. The best one is returned as the solution.

3.2. Dynamic entities

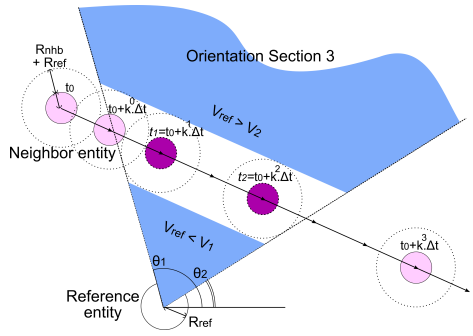


Figure 4: *Example of reachable space sectioning for a given time-interval. The resulting orientation section is characterized by several parameters: $t_1, t_2, \theta_1, \theta_2, V_1, V_2$.*

The objective of this first step is to compute a set of valid speed and orientation ranges for a given reference entity E_{ref}, with respect to the presence of neighbor moving entities. We describe our method for a single neighbor entity E_{nhb}, whilst the case of several ones is detailed Section 3.4.

Time discretization. We consider the environment state at $t = t_0$ and propose to search the E_{ref} reachable area for potential collision as explained previously. For that, we consider successively adjacent time-intervals having different durations: $[0, k^0\Delta t]$, $[k^0\Delta t, k^1\Delta t]$, $[k^1\Delta t, k^2\Delta t]$, $[k^2\Delta t, k^3\Delta t]$, etc. The $\Delta t > 0$ parameter defines the precision of the discretization, smaller being the best, and should correspond to the time needed by the entity to make one move. The $k > 1$ parameter is used to make the discretization non-uniform over the anticipated time, greater being the best, allowing the anticipation to be more precise in the near future than in the distant one. We use $\Delta t = 1$ and $k = 2$ in our model.

Reachable space sectioning. For each time-interval, we predict the E_{nhb} trajectory as a linear one and deduce the E_{ref} orientation range (*orientation section*) potentially leading to a collision with E_{nhb} as illustrated in Figure 4. We finally get as many sections as the number of time-intervals, each representing an orientation range for E_{ref}. The time-interval $[t_1\ t_2]$ used to compute each section is stored.

Critical speeds computation. For each orientation section, we compute the critical speeds V_1 and V_2 defined as follows: V_1 is the *maximal speed* allowed to avoid a collision by passing behind E_{nhb}; V_2 is the required *minimal speed* to avoid a collision by passing before (in front of) E_{nhb}. V_1 and V_2 have analytical expression, as solution to the following equation:

$$V_1 = \min_{t=t_1}^{t_2} \left((\left\| \overrightarrow{P_r P_n(t)} \right\| - R)/t \right)$$
$$V_2 = \max_{t=t_1}^{t_2} \left((\left\| \overrightarrow{P_r P_n(t)} \right\| + R)/t \right)$$

with $P_n(t) = P_n + \vec{v}_n t$ and where $\vec{v}_n$ is the E_{nhb} speed vector, P_r and P_n are respectively the positions of E_{ref} and E_{nhb} at $t = t_0$, and finally R is the sum of the bounding circles radii of the considered entities, eventually increased by a security factor to avoid strict contact cases.

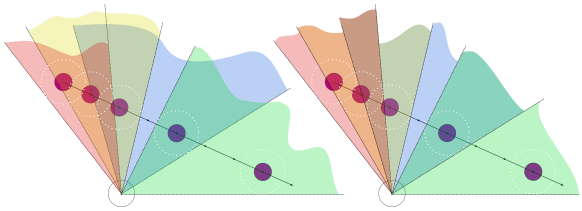


Figure 5: *The sectioning results in overlapping orientation ranges (left image). The merge of overlapping sections is easily done by subdividing orientation ranges in order to get adjacent sections (right image). The new characteristics of each subdivision are directly deduced.*

Merging several sections. By iterating previous computation for each time-interval, we get a set of orientation sections with their own characteristics (V_1, V_2, time-interval $[t_1\ t_2]$). A problem is to merge overlapping sections. Sections are finally subdivided in order to get a set of adjacent sections. Each new subsection characteristics are computed as follow: $V_{1new} = min(V_{1i}, V_{1j}), V_{2new} = max(V_{2i}, V_{2j}), t_{1new} = min(t_{1i}, t_{1j})$ and $t_{2new} = min(t_{2i}, t_{2j})$ with i and j indexing the two merged orientation sections. Note that if three or more sections overlap, this process can be reiterated successively considering pairs of sections until all are merged.

3.3. Static entities

The second step of the reactive navigation module consists of considering static obstacles. We handle static obstacles in approximately the same way than dynamic entities, but the problem is obviously simpler. In our environment database, obstacles borders are modeled as line segments. Let us consider the case of a single line segment in the vicinity of E_{ref}. Our first objective is to subdivide S_{obst} as shown in Figure 6. For that, we first compute P_0 the nearest point from E_{ref} belonging to S_{obst}. We define points P_1 and P_1' if existing so that the length $P_0P_1 = P_0P_1' = v_{ref}\Delta t$ where v_{ref} is E_{ref} speed, and Δt defined in the previous Section. Then, we define the point P_2 so that $P_0P_2 = k^1 v_{ref}\Delta t$, P_3 so that $P_0P_3 = k^2 v_{ref}\Delta t$, and finally P_4 so that $P_0P_4 = k^3 v_{ref}\Delta t$. This set of points is arbitrary, however, it allows us to evaluate the constrained speeds toward the obstacle with a more accurate precision near to E_{ref}.

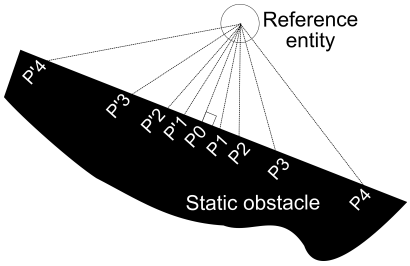


Figure 6: *Sections computation for static obstacles*

A set of adjacent orientation sections are computed as shown in Figure 6. We then compute the characteristics of each section in a same way than done previously with dynamic entities. However, V_1 is computed differently:

$$V_{1i} = \left\| \overrightarrow{P_iE_{ref}} \right\| / t_{2i}$$

where i indexes the considered orientation section. V_1 is thus the maximal speed at which E_{ref} can walk within the considered time-interval (depending on the considered section) without colliding the obstacle, analogically to the previous dynamic entity case. V_2 is here meaningless, and we set $V_2 = +\infty$.

3.4. Solving interactions

The third and final step of our reactive module consists of extracting a solution-move for the reference entity. The extraction is done in three successive steps. First, each orientation section previously computed is weighted using a cost function depending on sections characteristics. Second, superposed sections (related to different neighboring entities or obstacles) are merged (see Section 3.2) while accumulating costs. Third, the best section is used to compute the new speed and orientation.

Orientation section cost. The reference entity has to choose the best next speed and orientation according to the environment state and its goal. The function cost reflects the best choice among several criteria:

- Valid speed ranges (according to V_1 and V_2 of each section) must be close to the entity desired speed V_{des} and in its range of achievable speeds $[0; V_{max}]$.
- Orientation section limits $[\theta_1; \theta_2]$ must be as close as possible to the desired orientation θ_{des}
- Required accelerations to reach the new speed and orientations must be as limited as possible (limiting strictly them is not desirable because real humans are capable of important accelerations).
- The closer the section time-interval is in the future, the more confident we are in its cost.

The cost associated to speed variations and distance to desired speed is computed as follows:

$$C_{decel} = \begin{cases} 0 & if\ V_{des} \leq V_1 \\ 1 - \frac{V_1}{V_{des}} \end{cases} \qquad C_{accel} = \begin{cases} 0 & if\ V_{des} \geq V_2 \\ \frac{V_2 - V_{des}}{V_{max} - V_{des}} \end{cases}$$

$$C_{speed} = \alpha.min(C_{accel}, C_{decel})$$

where $\alpha \in]0;1[$ allows us to set a trade-off between speed changes and orientation changes. The cost associated to orientation changes (deviation) is computed as follows:

$$C_{dev} = (1-\alpha)\frac{1 - cos(\theta)}{2}$$

where θ is the minimum difference between the desired orientation and the orientation section limit angles. Note that $0 \leq C_{speed} + C_{dev} \leq 1$. According to t_1 the lowest bound to the time-interval of the concerned section, we finally compute a prediction confidence cost:

$$C_{pred} = 1 - \frac{t_1}{T + \beta} \qquad \text{with } 0 \leq C_{pred} \leq 1$$

where T is the maximal considered time for the prediction, and $\beta \in [0; +\infty]$ is a user-fixed parameter allowing us to get more or less confidence in predictions (this will change the pedestrian adaptation-time before a potential collision). The total cost of a given section is then:

$$C_{total} = (C_{speed} + C_{dev}).C_{pred}.C_{add}$$

where $0 \leq C_{add} \leq +\infty$ is an additive cost that can be introduced to take into account external factors, e.g., walking

close to another entity, preferential deviation to the left or the right, etc. Neutral value is $C_{add} = 1$.

Merging costs. Note, at this point, we have as many sectionings (whole sets of orientation sections) as the number of entities and static obstacles in the vicinity of E_{ref}. For each section of each set, we now have a cost. In exactly the same manner than presented in Section 3.2, we merge all the weighted sections. The cost of each subsection thus created is the sum of all the sections that were superposed and split to create it.

The fittest subsection has the lowest cost. In the corresponding range of valid speeds and orientations, we compute the closest to the ones desired by the reference entity, which is the final output to our reactive navigation module.

3.5. Discussion

Visibility of neighbor entities. As explained previously, our model predicts neighbor entities trajectories in order to decide on the best reaction to avoid them when necessary. We demonstrate in the next Section that real humans act in the same way, however, it is obvious they only do so for the humans they could visually perceive. In order to get a realistic reactive navigation, a perception field must be simulated. Our model distinguish two cases. When a neighbor entity is not seen because it is occluded by an obstacle, it is filtered out of the selection. As a result, an occluded neighbor entity has strictly no influence in the result. A limitation is the case were two pedestrians invisible one to the other converge toward the same place (e.g., at a street corner): they stop abruptly when finally perceiving one another, whereas a real human would anticipate this possibility and walks more carefully. When a neighbor entity is not seen because it is not in the field of view of the reference entity, we introduce it in the model, but having a null speed. Indeed, we consider that real humans feel someone is behind, but are unable to predict any trajectory. This also avoids backward motions provoking a collision with entities behind.

Connection with a locomotion animation module. As seen in Figure 1, output of our reactive navigation module is connected to a locomotion module. Our method may lead to important speed or orientation changes from which a realistic animation must be computed. In order to get as realistic animations as possible, we synchronize the animation module and the reactive navigation module, so that changes occur at the feet-land instants (left or right). Also, the locomotion module smooths variations itself.

4. Model Calibration and Validation

The previous Section described the technical basis of our model: resulting trajectories mainly depend on the parameters used in the cost functions (α, β). In order to get realistic behaviors, a calibration of the model is required. In previous works, such a calibration is achieved by analyzing video captures of real crowd motions. However, automatic analysis techniques are not always applicable due to lighting conditions, and analysis by hand requires great effort. Moreover, pedestrians goals are generally unknown in video sequences, which may prevent strict comparisons between real data and simulations. For these reasons, we prefer to use a motion capture system to collect our reference data, with protocols defining the goals of each participant.

In a first experiment, we measure the interaction phenomenon in the following situation: two pedestrians achieve navigation tasks in an empty environment, we force them to have more or less interacting trajectories and observe adaptations to avoid contact. Results allow us to demonstrate the need for prediction in a realistic reactive navigation model and to calibrate some crucial factors. Secondly, we attempt to reproduce some typical crowd navigation situations at a microscopic level, such as corridor following, gate crossing, X-crossing, etc. in order to validate the model behavior in more complex scenes. For that, we captured as many participants as possible executing navigation tasks in an environment made of obstacles.

4.1. A protocol for model calibration: interactions between two pedestrians

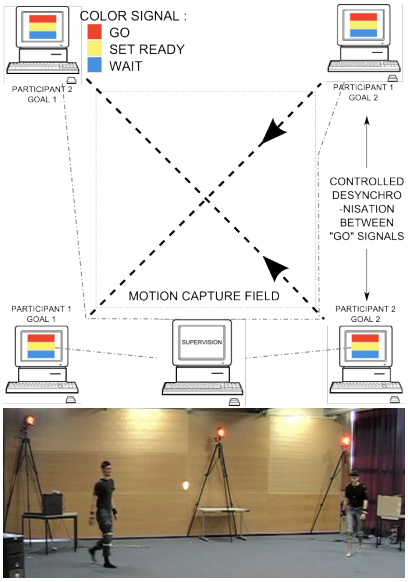


Figure 7: *Four computers displaying signals to participants controlling their start-time and goal. A motion capture system retrieves resulting interactions.*

The first experiment protocol allows us to control the interaction of two participants walking in an obstacle-free area, as illustrated in Figure 7. By interaction, we mean the

required trajectory adaptations made by each participant in order to avoid a collision. The objective of the experiment is to qualify the avoidance strategies developed by the participants given the conditions imposed by the protocol, and to quantify the trajectory corrections made in terms of velocity and orientation changes. For that, we place four computers at the corners of a square area visible for a motion capture system. Each participant must go from a given computer to the diagonally opposite one. We get a temporal control on the experiment by transmitting a start signal to participants using computer displays. All computers are synchronized in order to precisely control some delays between the start time of each participant, and to provoke more or less important interactions between them. We deduce the conditions for an interaction to occur or not. Participants always see each other but only perceive their own start signal. Finally, participants are equipped with 34 markers to get full body motion capture data. A total of 145 interactions were captured, 6 participants where involved, we placed computers in order to form $\pi/2$ or $\pi/3$ angles between trajectories.

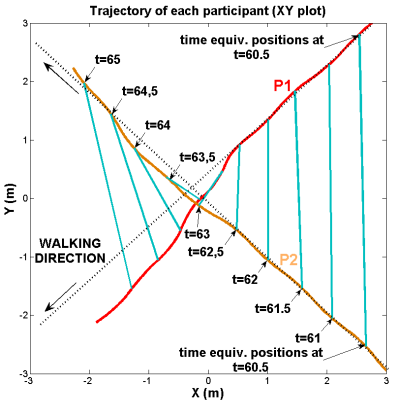


Figure 8: *Horizontal trajectory of 2 interacting participants.*

We detail our analysis method over a specific case whose results are shown in Figures 8 and 9. Here, as it can be seen in Figure 8 representing the horizontal trajectory of participants $P1$ and $P2$, a strong interaction occurred between experiment times 60s to 66s: trajectories are conspicuously deformed in order to avoid collision. We joined the respective positions of $P1$ and $P2$ at equivalent times (each half a second) in order to provide a temporal indication of events. A first look at the results leads one to think that participants reacted late (t=62.5s), just before collision: $P2$ passes before $P1$, $P1$ decelerates and turns to the left while $P2$ turns a little to the left to facilitate the passage.

But motion capture data allow a more precise analysis and the real intentions of participants appear clearly in Figure 9: top and bottom plots are the speed and orientation of each participant (respectively red and blue plots). In dashed horizontal lines, we represent some mean reference values for

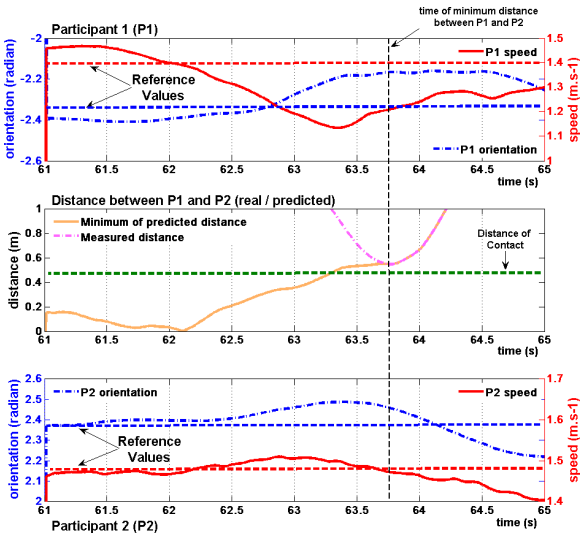


Figure 9: *Orientation and speed variations of participants $P1$ (top) and $P2$ (bottom). Comparing the predicted and measured distances between participants allows to detect the time of adaptation to avoid a collision (center).*

speed and orientation, measured during experiments where participants execute identical navigation tasks alone. We observe that two successive corrections finally compose this interaction:

- $t \in [61;62s]$: $P1$ has a higher speed and deviates to the right compared to mean reference values. This reveals her first intention to pass before $P2$, who has a normal behavior. But $P2$ naturally walks faster (looking at the reference mean values) than $P1$ and this first strategy fails.
- $t \in [62;62.5s]$: $P1$ decelerates.
- $t \in [62.5;64s]$: A combined reaction is now clearly visible: $P1$ increases its deceleration and deviates to the left, which will allow her to pass behind $P2$, who facilitates the success of this new strategy by accelerating and deviating to the left.
- $t > 64s$: After the time where distance between $P1$ and $P2$ is minimal, participants achieve their goal and no particular interaction is observable.

Previous analysis showed that corrections may appear early in the experiments ($P1$ for $t = 61$ to $62s$), and differences between reference and measured values may be minor and hardly detectable. We introduce a criterion to both qualify automatically situations of interactions in the different experiments and answer a crucial question: are corrections made by participants pertinent (i.e., does collision occurs if no adaptation is made)? This will allow us to conclude on the accuracy of participants in evaluating potential collisions. Figure 9 illustrates our criteria: the red curve is the measured distance between $P1$ and $P2$. The violet one is $MinDist_{pred}$

defined as follow:

$$Pred_{1,2}(t,u) = Pos_{1,2}(t) + (u-t)\overrightarrow{v_{1,2}}(t)$$

with t is the experiment time, $u \in [t;\infty]$ a parameter, $Pred_1(t,u)$ and $Pred_2(t,u)$ are respectively the predicted positions of $P1$, $P2$ in the future time u, according to their current position $Pos_1(t)$, $Pos_2(t)$ and speed vector $\overrightarrow{v_1}(t)$, $\overrightarrow{v_2}(t)$.

$$InterDistance_{pred}(t,u) = \| \overrightarrow{Pred_1(t,u)Pos_2(t,u)} \|$$

is the evolution of the distance between $P1$ and $P2$ according to the previously predicted positions.

$$MinDist_{pred}(t) = \min_{u=t}^{\infty} \left(InterDistance_{pred}(t,u) \right)$$

The plots reveal that the anticipation of $P1$ was accurate. Despite her initial faster speed and deviation to the right, she would not have been able to avoid $P2$ with a minimum predicted inter-distance below $0.2m$. Her change of strategy is revealed when $Mindist(t)$ is null at $t = 62.1s$ (from "passing in front of $P2$" to "passing behind $P2$").

By analyzing the whole set of experiments our first conclusions are:

- Reactions are observable for $MinDist_{pred}(t) < 0.5m$, which allows automatic distinction between cases where interaction occurs or not.
- Anticipation is up to several seconds before a potential collision. Our experimental conditions did not allowed us to determine an upper bound to anticipation time because of the size of the motion capture field.
- Reaction is a combination of speed and orientation adaptation. Deviations are bounded whereas decelerations can lead to a complete stop for one of the participants. This occurs especially when participants modify both their trajectory so that $MinDist_{pred}(t)$ remains below $0.5m$ (conflicting corrections). The closer the collision is in time, the more speed adaptation is preponderant.
- Interaction is an accurate phenomenon. If no collision is predicted by our criterion, no reaction appears in data. At the other extreme, participants detect interaction situations early.

We calibrate our model to synthesize interacting pedestrians trajectories in a realistic manner. Anticipation is set to 8 seconds. In order to find a trade-off between performance and precision, time is discretized in a non-uniform manner ($1s$ steps in the near future, up to $4s$ for the last period). Minimal and maximal velocities to avoid a collision are computed for a set of walking directions, allowing combination of speed and orientation variations to avoid other pedestrians. We tuned the cost function to fit experimental data in a pragmatic manner. Statistical analyses are still on-going in order to calibrate the model automatically from data. Moreover, we noticed individual factors and deeper analyses should allow us to determine a variety of individual profiles to calibrate our model.

4.2. A protocol for model validation: capturing microscopic crowd phenomenon

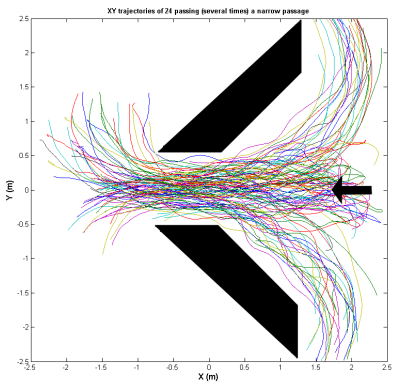


Figure 10: *A microscopic crowd phenomenon: crossing a narrow passage with funnel shape.*

In a second experiment, we capture 24 participants - each equipped with 5 markers placed on the chest - navigating in an area where we place static obstacles. The objective is to reproduce some frequently encountered situations where crowd flows meet, and to check for realistic emerging crowd behaviors in simulations (such as lane formations in corridors where opposite pedestrians flows meet). Figure 10 illustrates an example of obstacles setup, superposed with the set of recorded horizontal trajectories. Each participant crosses the narrow passage from the right to the left, and must circle the obstacle to reenter in the area from the right. We tested several setups: X-crossing with several flow directions repartitions, corridors with single or opposite flows, gate crossing, etc. We can draw first conclusions from comparisons between simulations and real data. Characteristic phenomenons could emerge from our model as observed in real data such as lane formation in corridors where two opposite flows meet, pedestrians going in same directions at X-crossing tend to group in order to facilitate their passing-by, high speed changes occur where flows meet with visibility limited by obstacles, etc. Thus, we validate qualitatively the basis of our model. However, quantitatively, simulation and real data still need to be accurately compared, and we are currently performing a deeper analysis.

4.3. Discussion

From real data, we can discuss the limitations of our model. During the calibration step, we found a balance between the cost of direction changes and speed changes (α parameter of the cost function Section 3.4) that allowed us to have similar results in simulation and experiments with identical conditions. However, we could not find a calibration that satisfies all situations. Simulation results were still valid (no collision, and no dead-locks), but realism was decreasing in some

specific situations. We think an always-valid calibration is impossible to find for two reasons. First, our anticipation time, T, is a constant parameter, whereas real humans act in a more reactive way where population is dense. Second, time-interval duration step, Δt, is also a constant parameter during the sectioning step (see Section 3.2). In other words, our model precision is constant and should depend on the situation. We believe that these two model parameters should be adapted according to local population density (at least) in order to improve the realism of the results.

5. Results

Performance We have run a variety of situations reproducing the ones we motion captured, as shown in the accompanying video. Implementation is in C++ and simulations ran on a Pentium M $1.6GHz$, $1GB$ memory, with an Nvidia 4200Go 64MB graphics card.

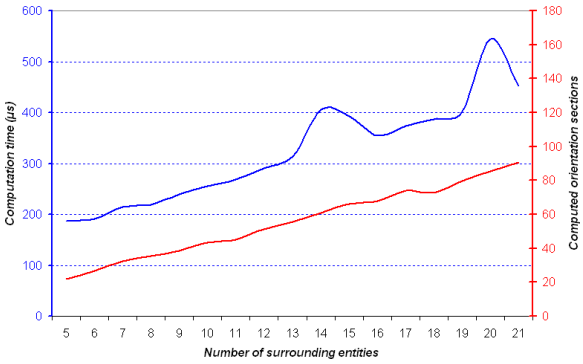


Figure 11: *Solution Performance. The blue curve plots the computation time required for solving* one *interaction according to the* number of neighbor entities *taken into account. The red curve plots the number of orientation sections created in order to compute the solution. From these plots, it is possible to compute global performances with respect to the density of people.*

The performance provided in Figure 11 does not correspond to the computation time functions of the total number of simulated entities, but to the computation time for one entity according to the density of people. In fact, this density of people implicitly defines the number of neighbor entities to take into account for the computation. Moreover, the plot limit of 21 surrounding entities has not been specified, but is a result of our benchmarks using by far more entities in different situations (an average of 100 moving agents, plus the walls). Then, the computation time for one entity, between 200 and 500 μs, may appear high compared to previous approaches, but is relatively stable for growing entities numbers. In addition, our model is only refreshed at each foot step, corresponding to an average rate of $1-2Hz$. Thereby, taking the worst cases, our model can easily handle 1,000 entities in real time, and its complexity is scalable and could be reduced to simulate more entities by decreasing the anticipation time. However, as previously mentioned, this model is integrated in a whole virtual human architecture, managing other tasks like path planning, rational behavior, animation, and so on. Based on our experiments, we are able to fully simulate and animate approximately 150 entities in real time. Moreover, the performance depends on the complexity of the environment, and on the density of the crowd, that is why it is difficult to provide representative performance considering the number of influent parameters.

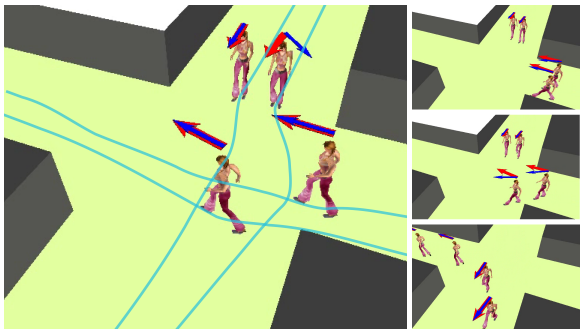


Figure 12: *Detailed view of a unit interaction case. Red arrows are desired directions θ_{des} while blue ones result from the reactive navigation. Virtual humans adapt both their speed and orientation in quite an optimal and realistic manner to avoid each other (see video for animation).*

Realism As mentioned Section 4.2, we could compare our simulations with real data, however, a quantitative validation of our model is still in progress. Our first results are promising. We solved several drawbacks observed in previous approaches. First, anticipated reactions in low density areas improve results realism (Figure 12): in such cases, conflicting states (typically, close entities face to face) never appear in simulations. Indeed, thanks to a sufficiently stable state of the environment, model predictions remain valid for a large enough interval to always avoid any close interaction. Second, in dense population situations, pedestrians do not have jerky trajectories with successive and contradictory reorientations or turn-backs. In this case, next-future predictions are accurate enough to avoid such unrealistic behaviors and dead-locks. However, distant time predictions appear to be useless because of the constantly changing situation.

6. Conclusion and Perspectives

We presented a novel approach to solve interactions between virtual pedestrians in the context of realistic crowd simulations. Our first contribution is to introduce long-time predictions in our model when accounting for other pedestrians moving around. Such a prediction allowed us to solve major drawbacks of previous approaches to this known difficult problem. Our second contribution is to propose experiments

to validate and calibrate our approach using a motion capture system. Compared to video-streams analysis, we reach a higher level of precision that allowed us to obtain a fine analysis of events occurring during real humans interactions. We also captured crowd behaviors at a microscopic level in order to validate our simulation results. Work is still in progress concerning this last point.

Our results are promising and we can identify our model limitations to get even more realistic results: this determines our future work directions. First, we want to dynamically adapt the model parameters to the variations of environment context. Especially, the prediction time-ranges and precision must fit the local population density. Second, the model does not account for social factors. We want our model to be able to consider couples of pedestrians or larger groups navigating among other pedestrians: such groups shall remain as gathered as possible during their navigation. We believe our implementation capable of supporting such evolutions easily, and work is underway. Finally, connections to the animation module must be enhanced. A feedback must emanate from the locomotion module in order to score the proposed reactions, purely in terms of realism of motion. This additive cost would help in taking realistic decisions. Recent results in animation evaluation techniques is an inspiration source to reach such a goal.

7. Acknowledgement

This work is funded by the French Research Agency ANR Project ANR05RNTL02501 *SIMULEM*, in collaboration with the French railroad company *SNCF*. We also thank members of our team who provided the animation module used in this work, *MKM* [KMA05], and who helped us during the motion capture experiments.

References

[BLA02] BAYAZIT O. B., LIEN J.-M., AMATO N. M.: Roadmap-based flocking for complex environments. In *Pacific Conf. on Comp. Graphics and Applications* (2002), pp. 104–113.

[BMdOB03] BRAUN A., MUSSE S., DE OLIVEIRA L., BODMANN B.: Modeling individual behaviors in crowd simulation. In *16th Int. Conf. on Computer Animation and Social Agents* (New Brunswick, NJ, USA, May 2003), IEEE, pp. 143–148.

[Bol81] BOLES W.: Planning pedestrian environments: A computer simulation model. *Man-Environment Systems 11* (1981), 41–56.

[GKM*01] GOLDENSTEIN S., KARAVELAS M., METAXAS D., GUIBAS L., AARON E., GOSWAMI A.: Scalable nonlinear dynamical systems for agent steering and crowd simulation. *Computers and Graphics 25*, 6 (2001), 983–998.

[Gof71] GOFFMAN E.: *Relations in public : microstudies of the public order*. New York : Basic books, 1971.

[Hen71] HENDERSON L. F.: The statistics of crowd fluids. *Nature 229* (1971), 381–383.

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

[KMA05] KULPA R., MULTON F., ARNALDI B.: Specific representation of motions for interactive animation with several characters. *Computer Graphics Forum 24*, 3 (2005), 343–352.

[LKF05] LAKOBA T. I., KAUP D. J., FINKELSTEIN N. M.: Modifications of the helbing-molnár-farkas-vicsek social force model for pedestrian evolution. *Simulation 81*, 5 (2005), 339–352.

[LMM03] LOSCOS C., MARCHAL D., MEYER A.: Intuitive behavior in dense urban environments using local laws. In *Theory and Pratice of Computer Graphics* (2003).

[MT01] MUSSE S. R., THALMANN D.: Hierarchical model for real time simulation of virtual human crowds. *IEEE Transaction on Visualization and Computer Graphics 7*, 2 (jun 2001).

[PDB05] PARIS S., DONIKIAN S., BONVALET N.: Towards more realistic and efficient virtual environment description and usage. In *First International Workshop on Crowd Simulation (V-Crowds'05)* (2005), VRlab, EPFL.

[PDB06] PARIS S., DONIKIAN S., BONVALET N.: Environmental abstraction and path planning techniques for realistic crowd simulation. *Computer Animation and Virtual Worlds* (2006), 325–335.

[Pes71] PESCHL I.: Passage capacity of door openings in panic situations. *BAUN 26*, 2 (1971), 62–67.

[PM78] PREDTECHENSKII V. M., MILINSKII A. I.: *Planning for foot traffic flow in buildings*, National Bureau of Standards ed. Amerind Publishing CO Pvt Ltd, New Dehli, India, 1978.

[Rey00] REYNOLDS C. W.: Interaction with groups of autonomous characters. In *Game Developers Conference* (2000).

[SGC04] SUNG M., GLEICHER M., CHENNEY S.: Scalable behaviors for crowd simulation. *Computer Graphics Forum, Eurographics'04* (2004).

[ST05] SHAO W., TERZOPOULOS D.: Autonomous pedestrians. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM Press, pp. 19–28.

[TCP06] TREUILLE A., COOPER S., POPOVIĆ Z.: Continuum crowds. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM Press, pp. 1160–1168.

[Yam98] YAMORI K.: Going with the flow: Micro-macro dynamics in the macrobehavioral patterns of pedestrian crowds. *Psychological Review 105*, 3 (1998), 530–557.

Symposium on Computer Animation 2009 (SCA '09) (2009)
(Editors)

Experiment-based Modeling, Simulation and Validation of Interactions between Virtual Walkers

Julien Pettré[1] Jan Ondřej[1] Anne-Hélène Olivier [2] Armel Cretual [2] Stéphane Donikian[1]

[1]Bunraku team, IRISA / INRIA-Rennes, France
[2]M2S, UEB, Université de Rennes 2, France

Abstract

An interaction occurs between two humans when they walk with converging trajectories. They need to adapt their motion in order to avoid and cross one another at respectful distance. This paper presents a model for solving interactions between virtual humans. The proposed model is elaborated from experimental interactions data. We first focus our study on the pair-interaction case. In a second stage, we extend our approach to the multiple interactions case. Our experimental data allow us to state the conditions for interactions to occur between walkers, as well as each one's role during interaction and the strategies walkers set to adapt their motion. The low number of parameters of the proposed model enables its automatic calibration from available experimental data. We validate our approach by comparing simulated trajectories with real ones. We also provide comparison with previous solutions. We finally discuss the ability of our model to be extended to complex situations.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation; I.6.4 [Simulation and Modeling]: Model Validation and Analysis; I.6.5 [Simulation and Modeling]: Model Development;

Keywords: steering method, collision avoidance, interaction

1. Introduction

The computer animation community put in a great deal of effort to provide virtual humans with autonomy of locomotion. Despite the apparent simplicity of this everyday task, simulating locomotion in a realistic manner is complex, especially when virtual walkers are moving in environments made of many static and dynamic obstacles. A large body of prior work suggests that simulating interactions between virtual walkers in a realistic manner is particularly difficult.

An interaction occurs between walkers when a reciprocal influence is observed on their respective trajectory: each one adapts its own motion in order to avoid the others. Understanding and simulating interactions between humans is complex due to the possibly high number of factors involved. Human locomotion is generally driven by a goal to reach, while it is constrained by physical and biomechanical factors. In addition, environmental factors - such as the presence of obstacles - set supplementary constraints. Trajectory adaptations are based on the perception humans have of others' motion, which is naturally error-prone. These first two kinds of factors, related to physics and perception, are obviously important during human interactions, but secondary factors also need attention. First, sociological and cultural factors influence human reaction according to some tacit rules (deviating preferably to the left or to the right, avoiding elderly people more carefully, etc.). Psychological factors are also involved: people walk according to their mental state (hurrying, wandering). They are capable of moving in an expressive manner to objectify this internal state.

Our motivation is to achieve realistic simulation of interactions between walkers. Our objective is first to better understand how real humans behave in such situations. Our results allow us to discuss several assumptions which were formulated to build existing computational models (developed by both the crowd simulation and computer animation communities). We elaborate a new model that better fits our experimental observations. Our approach is however based on two major assumptions. First, a complex interaction that involves several walkers simultaneously can be described as a

combination of pair-interactions involving two walkers only. Second, physical and perceptual factors are preponderant. Secondary factors are then to be taken into account in future works. As a result, we first focus our study on interactions to the two-walker case only. We however describe how the proposed model can handle multiple interactions. Understanding how humans combine pair-interactions to solve complex ones is discussed in this paper, but certainly deserves a complete study. Our contribution is threefold. Our primary contribution is a model for solving interactions between virtual walkers, presented in Section 4. The model is elaborated from experimental observations of real interactions between walkers, which sets our second contribution. Our experimental data are made available to the research community. Our third and final contribution is to propose a quantitative evaluation of our model results. This evaluation enables an objective assessment of simulated trajectories, as compared with real data, or ones resulting from previous solutions.

The remainder of this paper is organized as follows: in Section 2, we review prior work in modeling interactions. Section 3 describes our experimental results. The model we elaborated from experiments is described in Section 4. Comparison of our model results with previous techniques is detailed in Section 5. Finally, we discuss limitations and possible extensions of our model to answer complex situations in Section 6, before concluding.

2. Related Work

Mainly three research fields addressed the problem of interactions between walkers. First, cognitive sciences studied the influence of obstacles (static or moving) on human locomotion. Their experiments demonstrated that humans combine notions of time, distance and velocity to avoid collisions. The second field focused on crowd simulation. Crowd simulators mainly aim at studying the impact of numerous human-human and obstacle-human interactions on the global circulation of many walkers. Their main objective is to achieve realistic simulations at a macroscopic level, even though solutions are often based on microscopic models. The third field is Computer Animation which needs for believable individual locomotion trajectories and developed specific approaches to simulate interactions. Two main classes of solutions exist. First ones are based on steering methods: they are general and efficient, however, evaluating their level of realism is still an open question. The latter class of solutions relies on a database of captured real interactions, reused in simulations to imitate how real humans solve interactions. A high level of realism is intrinsically obtained; nevertheless, these solutions' validity domain is limited to the database content.

Time-to-collision and personal space. Avoiding collisions is a spatiotemporal problem. Cognitives sciences divided its temporal and spatial dimensions into two different notions: the time-to-contact TTC, and the personal space. According to Cutting and colleagues [CVB95], humans avoid collisions by answering two successive questions: will a collision occur? When will this collision occur? Answers result from the visual perception of their environment and of the moving or stationary obstacles. Lee [Lee76] and Trésilian [Tré91] demonstrated that the optical flow generated from the visual perception of a moving object is sufficient to directly evaluate TTC. The real nature of information used by humans to evaluate TTC is still an open question; however, humans adapt their motion to avoid collisions in order to preserve admissible TTC. Velocity, distance and time are intrinsically linked together. As a result, TTC can also be interpreted as a preserved distance between humans and obstacles, giving rise to the *personal space* notion. Personal space can be defined as a safety area preserved by walkers around them. The personal space gives walkers enough time to react to an unexpected moving obstacle appearing in their perception field. Gérin-Lajoie and colleagues [GLRM05] experimentally measured the personal space's shape and dimensions. They found the personal space is elliptic, as intuitively imagined by Goffman [Gof71]. The novelty of this study is to focus on personal space measurement while moving. However, the experimental process was based on the interaction between a human walker and a moving manikin mounted on an overhanging rail.

Reactive approaches. Solving interactions is certainly a crucial component of crowd simulation. Helbing's social forces model is probably the most popular approach [HM95]. The model was later revisited and calibrated for specific situations [HBJW05], or integrated into a software platform in order to solve well-known artifacts [PAB07]. In this model, virtual walkers are modeled as velocity-controlled particles undergoing a sum of acceleration forces with an analogy to Physics. Interactions are modeled as repulsive forces between walkers, and expressed as a function of their relative distance. Treuille and colleagues [TCP06] also make an analogy to Physics, but formulate interactions as a minimization problem. Walkers' motion is deduced from a potential field, whose dynamic component results from a repulsion emitted by walkers. Walkers avoid each other implicitly, interactions are not explicitly modeled. In [HLTC03], interactions between humans are modeled as a mass-spring-damper system: stiffness and viscosity terms change with respect to relative distance between walkers.

Anticipated collision avoidance. The *steering behaviors* introduced in [Rey99] enable interaction solving with anticipation. The *unaligned collision avoidance* behavior extrapolates walkers' trajectories - assuming that their velocity is constant - and checks for collisions in a near future. A reactive acceleration is computed for both walkers, in the direction opposite from the one of the future colli-

sion. Van den Berg and colleagues extend the *Reciprocal Velocity Obstacle* principle from Robotics [vdBPS*08]. Similarly to Reynolds' steering, this technique enables collaborative interaction solving with anticipation. Finally, Paris and colleagues [PPD07], inspired by Feurtey [Feu00], solves the problem from an egocentric perspective (i.e., walker-centered). In this approach, perceived neighbors' motion is also linearly predicted; an admissible velocity domain for each walker is deduced. A cost function is used to compute a specific velocity command belonging to the admissible ones. More recently, Kapadia and colleagues proposed an egocentric anticipative model in [KSHF09].

Imitating humans. Several solutions appeared in the literature taking advantage of motion capture or video tracking technologies to create databases of real interactions [MH04]. In [LCHL07], relative motions and positions between virtual humans are related to behaviors: this approach applies to the collision avoidance problem, but more generally enables behavioral crowd animation. In [LCL07], the authors solve interactions occurring in a simulation by retrieving the most similar example from the database; however, controllability and efficiency problems rise. Both of these problems are solved in [TLP07]: walkers are described using a state-vector, whilst a captured motion is modeled as a state-vector change. Given a user-defined state command, a motion sequence is found to reach the desired state in a near-optimal manner. Some components of the state-vector are used to describe interactions with one neighbor walker: this technique is thus able to solve interactions.

Our approach. The experimental study proposed in the next section allows us to describe how humans solve collisions. We demonstrate that the adaptations are not purely reactive and cannot only be modeled as a function of the distance between them. It is however possible that this assumption becomes true in the case of crowded areas where walkers have numerous and intensive interactions. Nevertheless, a realistic simulation and animation of virtual walkers in the general case need anticipation. We have mentioned existing solutions to anticipate a reaction. However, several questions remain: some approaches anticipate a reaction at constant distance or time to collision, others immediately when interaction is detected. Our experiments demonstrate that interactions start with an observation period of time, which allows humans to estimate other's motion accurately enough before reacting. Our model accounts for perception-errors in order to evaluate when reactions occur. Moreover, previous solutions assume velocity is constant before interaction, which enables linear extrapolations of trajectories. We discuss and address the more complex case where walkers are accelerating when interactions are initiated. Concerning imitation techniques, their main advantage is their intrinsic realism. However, two main drawbacks limit their application. First, efficiency does not always enable real-time simulation. Second, their validity domain is restricted by the content of example databases. Furthermore, some of these techniques do not apply to multiple interactions. About steering methods, the obtained level of realism has not always been evaluated. Brogan and Johnson proposed evaluation metrics to assess simulated trajectories [BJ03]. Singh and colleagues [SNK*08] proposed a framework to evaluate the ability of steering methods to address interactions among obstacles. We propose an objective evaluation of our results based on real data.

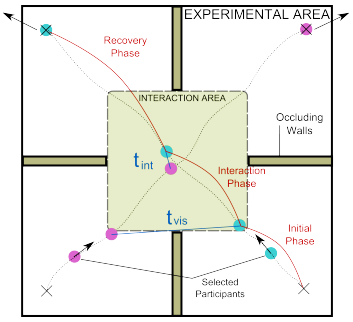


Figure 1: *Illustration of the proposed experimental protocol.*

3. Experimental Study

Objectives Our objective is here to describe how humans solve pair-interactions. We choose to observe interactions under protocol-controlled conditions, in order to diminish - and more important, to maintain constant between each experimental sample - the role of secondary factors: we focus our attention on physical and perceptual factors only. Accurate measures of motion adaptations are desired, and we choose to use a motion capture system to acquire experimental data.

Protocol The proposed experimental protocol is illustrated in Figure 1. At the start of each experiment, 4 participants stay still at each corner of a square *experimental area*. We randomly give 2 of the 4 participants the simultaneous order to walk toward the opposite corner (along each diagonal), the 2 others leave the experimental area. Start signals are given to participants by network-synchronized computers. Participants have orthogonally intersecting paths and synchronized trajectories: they are likely to interact, but not necessarily. Initial conditions of interactions change for each experiment because participants are asked to walk at their own comfort speed. Occluding walls prevent participants to observe each other before reaching their comfort speed. Our experimental square is $15m$ long, interaction area is $10m$ long. We randomize the selection of participants, so that they cannot anticipate the direction from which one may appear. 30 subjects have taken part in this experiment. We recorded 429 experimental samples.

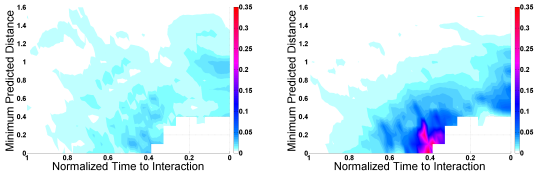


Figure 2: ***left:*** *average adaptation over all experiments made by participant passing first, function of minimum predicted distance and normalized time to interaction.* ***right:*** *average adaptation over all experiments made by participant giving way, function of minimum predicted distance and normalized time to interaction.*

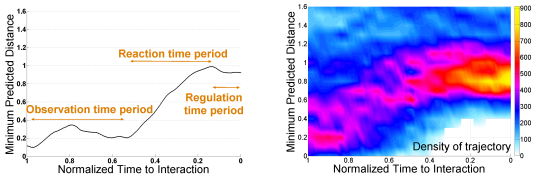


Figure 3: ***left:*** *minimum predicted distance for one experiment, function of normalized time to interaction. Three successive periods of time are observed: observation, reaction, and regulation.* ***right:*** *density of minimum predicted distance trajectories over all experiments, function of normalized time to interaction*

Method The trajectory was established from the mean of the two shoulder markers: $P(x,y)$. Velocity is noted $V = dP/dt$ and acceleration is noted $A = dV/dt$. We note θ the velocity vector direction, and v its norm. We check that lateral velocities can be neglected, and assume locomotion is non-holonomic in our case [ALHB08]. Data are filtered to remove noise and reduce the effect of natural oscillations (Butterworth low-pass second order filter, $1Hz$ cutoff frequency, zero phase shift). Trajectories are decomposed into three periods of time: participants start walking during the *initial phase*, during which they reach their comfort speed. The *interaction phase* starts when participants are able to see each other (with respect to occluding walls), at time $t = t_{vis}$. The time when the distance between participants is minimal is called the *interaction time* t_{int}. Finally, for $t > t_{int}$, participants head again for their goal during the *recovery phase*. Our study is focused on the *interaction phase*, for $t_{vis} < t < t_{int}$. In the absence of interaction, participants have constant velocity inside the interaction area. Their trajectory can be predicted linearly as follows:

$$\tau_{pred}(t,u) = P + (u-t)V, \tag{1}$$

τ_{pred} is the predicted trajectory from instantaneous position and velocity at time t. Parameter $u > t$ corresponds to the *future time*. For any time t belonging the interaction phase, we are able to predict the distance at which they would meet if no adaptation is made (the minimum predicted distance mpd):

$$mpd(t) = \min_{u>t} \|\overrightarrow{\tau_{pred,1}(t,u)\tau_{pred,2}(t,u)}\|, \tag{2}$$

where $\tau_{pred,1}$ and $\tau_{pred,2}$ are the predicted trajectories for each of the participants at time t. As participants walk at their own comfort speed, initial interaction conditions change for each experiment. This results in a variety of initial $mpd(t = t_{vis})$ values over all our experiments. Finally, in order to enable direct inter-experiments comparison, we normalize the duration of the interaction phase and define the normalized time-to-interaction tti_n as follows (average duration of interaction phase over all experiments is 4 s.):

$$tti_n(t) = (t_{int} - t)/(t_{int} - t_{vis}), \tag{3}$$

tti_n ranges from 1 to 0, respectively corresponding to the normalized time at which the interaction phase starts, and next, ends.

Minimum predicted distance as a motion adaptation criterion. As participants' motion is linear in absence of interaction, we simply detect motion adaptation by direct measuring of accelerations. We define thus the adaptation quantity $a(t)$ as follows:

$$a(t) = \|A(t)\| = \|\dot{V}(t)\| = \sqrt{\dot{v}^2(t) + \dot{\theta}^2(t).v^2(t)}. \tag{4}$$

Previous studies state that walkers first determine whether a collision will occur, and react (or not) accordingly. As a result, adaptations are to be detected for low mpd values only. Figure 2 plots participants' adaptation averaged over all our experiments, in function of mpd (on vertical axis) and tti_n (horizontal axis). Adaptations of the participant passing first and of the one giving way are plotted separately (respectively the top and bottom plot). A strong relation between adaptation and mpd is effectively observed: when $mpd > 1m$, almost no acceleration is detected. Note that adaptations at low normalized time values ($tti_n < 0.1$) correspond to the initiation of the recovery phase.

Three successive stages. The left plot of Figure 3 shows the evolution of mpd in time for one single experiment. At the beginning of interaction phase, we observe $0.1m < mpd < 0.35m$: collision is predicted (mpd is effectively a body center-to-center distance). mpd remains low for half of the interaction phase. We call this first stage the *observation period*. Following this, mpd is increased during the *reaction period* to an acceptable value of $1m$, enabling collision avoidance: participants necessarily adapt their motion to increase mpd. However, we cannot determine from this plot only if adaptation is made by one participant or both. Finally, mpd is maintained during the *regulation phase*. mpd gives a clear temporal description of interactions. In order to obtain a statistical overview of interaction solving, we cumulate mpd trajectories for all experiments, and plot the density of trajectories: the result is given at the right of Fig-

ure 3. Note that we later use the *density plot* to qualitatively compare simulated and real data. We conclude that the observation period statistically takes place for $1 > tti_n > 0.8$, which means during the first $0.8s$ of the interaction phase. Reaction period averagely lasts for $1.6s$ ($tti_n \in [0.8, 0.4]$). Finally, during the remaining time ($1.6s$ until $tti_n = 0$, i.e., $t = t_{int}$) motion is regulated to maintain admissible mpd values. Note that mpd becomes closer to the distance between participants when the interaction phase is ending. As a result, we can see that the average distance at which participants meet is distributed around $0.8m$. The existence of the regulation period of time confirms interactions are solved with anticipation.

A role-dependent adaptation. Figure 2 separately plots the adaptations made by each of the two participants. We observe that both make adaptations, however, the first participant passing is clearly making less efforts. In conclusion, interaction is solved collaboratively, but asymmetrically. Further analysis also reveals different strategies: the first participant mainly adapts his velocity, whereas the one giving way combines velocity and orientation adaptations. This asymmetry confirms the notion of the personal space: in order to preserve this space, the participant giving way needs to make larger avoidances.

Adaptation is error-prone. A common experience we all have had while walking is to get close to colliding with someone after successive hesitations on how to avoid him. We could observe such hesitations: $mpd(t)$ value is lowered in time instead of being increased. Antinomic adaptations occur only when mpd is initially close to $0m$. In such a case, the role of each participant is not clearly predictable. The observation period then becomes longer, and reaction is delayed. Do participants refine their motion estimation to determine their role? Nevertheless, such cases provoke the acceleration peak that can be seen in Figure 2, right plot, for $tti_n \approx 0.4$ and $mpd \approx 0m$. The density plot however reveals that such cases remain rare.

4. A model for solving interactions between walkers

We elaborated a model for solving interactions between virtual walkers from our experimental observations. Our model is based on an egocentric representation of walkers' relative motion. In the following part, we describe how our model solves pair-interaction. Next, we describe a calibration technique to compute realistic model parameters from real interaction data. Finally, the case of multiple interactions is addressed in Section 4.3.

4.1. A Model for Solving Pair-Interactions

Overview. An interaction is solved for each of the two involved virtual walkers independently. Walkers are modeled

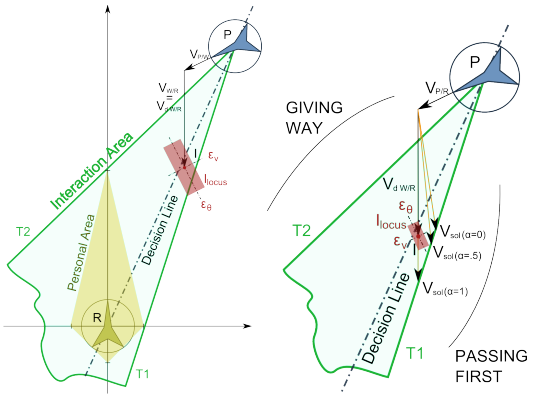


Figure 4: ***left:*** *Illustration of the components of the proposed model. Solution is based on an egocentric representation of the interaction situation.* ***right:*** *Step 2 and 3 of a Model Iteration: walker's role is deduced from the position of I relatively to the decision line. A solution velocity is computed in order for I to exit the interaction area.*

as velocity-controlled moving points. Our description is supported by the example introduced in Figure 4, left. Two walkers walk straight forward at comfort velocity toward their goal, their trajectories are secant. We note $\mathcal{R}$ the reference walker for which the model is controlling the motion, displayed at the bottom of the figure. The perceived walker on the top-right of the figure is noted $\mathcal{P}$. Our approach is based on several components described below.

Model components The first step of our solution consists in computing the egocentric representation of the interaction situation, as illustrated in Figure 4. We first compute $\mathcal{P}$'s position and velocity relatively to $\mathcal{R}$. We consider thus the local coordinate system centered and oriented on the reference walker $\mathcal{R}$. $\mathcal{P}$'s relative position is noted $P_{\mathcal{P}/\mathcal{R}}$, and relative velocity is computed as follows:

$$V_{\mathcal{P}/\mathcal{R}} = V_{\mathcal{P}/\mathcal{W}} + V_{\mathcal{W}/\mathcal{R}}, \tag{5}$$

where $V_{\mathcal{P}/\mathcal{W}}$ is the velocity of $\mathcal{P}$ relatively to the world $\mathcal{W}$, and $V_{\mathcal{W}/\mathcal{R}}$ the relative motion of $\mathcal{W}$ relatively to $\mathcal{R}$ (simply deduced from absolute velocity vector: $V_{\mathcal{W}/\mathcal{R}} = -V_{\mathcal{R}/\mathcal{W}}$).

The reference walker's *desired velocity* V_d is directly deduced from its goal. V_d is oriented toward this goal, its norm is the reference walker's comfort velocity v_c. The constant v_c is thus an individual parameter. V_d is then expressed in the local coordinate system as a desired world velocity relatively to the reference walker: $V_{d\mathcal{W}/\mathcal{R}} = -V_d$. In the case presented in Figure 4, the reference walker is walking at the desired velocity as no adaptation has been made yet. As a result, desired velocity and current velocity coincide.

A *personal area* is set around the reference walker. Personal area has a kite shape. The kite approximates the elliptic shape experimentally measured by [GLRM05], but is

mathematically much simpler to handle. The kite allows the reference walker to keep more space available in front of it than in its back and on its sides. Note that the personal area here maintains a center-to-center distance between walkers. The kite's dimension on each side and on the back is $0.8m$ and velocity-dependent in front of it (where u_t is the unit of time):

$$l = 0.8 + 0.4v.u_t. \tag{6}$$

T_1 and T_2 are the tangents to the personal area passing by $P_{\mathcal{P}/\mathcal{R}}$. These tangents delimit an area called *interaction area* (colored in light green in Figure 4). We also define the *interaction point I* as follows:

$$I = P_{\mathcal{P}/\mathcal{R}} + u_t V_{\mathcal{P}/\mathcal{W}} + u_t V_{d\mathcal{W}/\mathcal{R}}. \tag{7}$$

Our technique is mainly based on the position of I relatively to the interaction area: virtual walkers play on this position to solve an interaction. I results from the relative motion of $\mathcal{P}$ to $\mathcal{R}$, however, each virtual walker can only act on the self-motion component of this relative velocity. Our experiments demonstrate the importance of motion perception to explain the interaction temporal structure. As a result, we model the interaction point location relatively to $\mathcal{R}$ as prone to errors. We only take into account velocity and orientation perception errors (respectively noted ε_v and ε_θ), as they result from position integration in time since interaction initiation. We neglect position and self-motion perception errors. We model perception errors by transforming the I point into a locus: I_{locus}. Errors are dependent on the observation time since the reference walker is perceiving $\mathcal{P}$: the longer the observation time, the lower the errors. We compute ε_v and ε_θ, as functions of time-to-interaction tti, as follows:

$$tti = \arg\min_k \|P_{\mathcal{P}/\mathcal{R}} + k.V_{\mathcal{P}/\mathcal{R}}\|, \tag{8}$$

$$\varepsilon_v = \beta_v.(1 - (\frac{t_{int}}{t_{int} + tti})^{\gamma_v}), \tag{9}$$

$$\varepsilon_\theta = \beta_\theta.(1 - (\frac{t_{int}}{t_{int} + tti})^{\gamma_\theta}), \tag{10}$$

where t_{int} is the time elapsed since $\mathcal{P}$ is perceived. β_v, γ_v, β_θ and γ_θ are parameters to be calibrated from real data as explained in the last paragraphs of this section. Default values are: $\beta_v = 0.5$, $\beta_\theta = 0.5$, $\gamma_v = 0.25$, $\gamma_\theta = 0.25$. We model I_{locus} as a rectangle aligned on the $\mathcal{P}$'s velocity vector $V_{\mathcal{P}/\mathcal{W}}$. Its length (in the direction of $V_{\mathcal{P}/\mathcal{W}}$) is ε_v and its width (orthogonally to $V_{\mathcal{P}/\mathcal{W}}$) is ε_θ (see Figure 4).

Finally, we define the *decision line* which joins $\mathcal{R}$ and $\mathcal{P}$. We now describe the successive steps of one model iteration.

Model Iteration Our model works according to the three following steps.

Step 1 - Is adaptation required? If I_{locus} is fully contained in the interaction area, then $\mathcal{R}$ has an accurate enough estimation of $\mathcal{P}$'s motion to be sure they will pass at a too low distance. Indeed, the extrapolated $\mathcal{P}$'s trajectory is crossing $\mathcal{R}$'s personal area. We are then in the situation illustrated by Figure 4, right image. (a detailed-view of Figure 4 where $\mathcal{R}$ is not represented anymore). The goal of the following steps is to move I on the limits of the interaction area. In the opposite case, the model iteration stops here.

Step 2 - What is the reference walker's role? By the interaction point definition (cf. Equation 7), the reference walker can modify I's position by adapting its desired velocity, which becomes the solution velocity. In order to solve the interaction in an optimal manner, I has to be on the limit of the interaction area: $I \in T_1$ or $I \in T_2$. These two solution domains respectively correspond to two different roles: passing first or giving way. Decision is taken from the relative position of I to the decision line. In the case of Figure 4, right, I is on the side of T_1.

Step 3 - Computing a solution velocity. We want $I \in T_1$ (T_2 when giving way), the solution velocity $V_{sol\mathcal{W}/\mathcal{R}}$ has to verify the following condition:

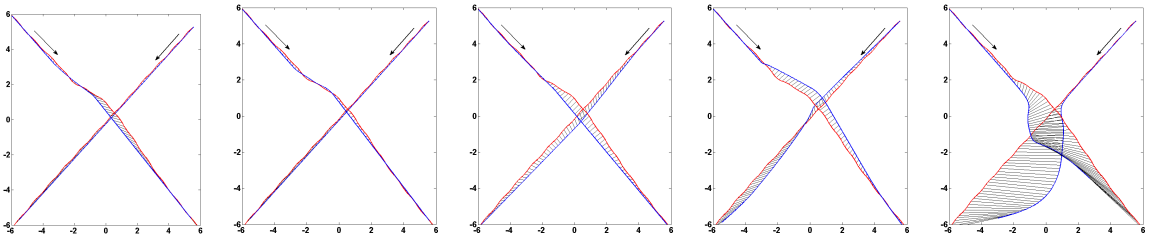
$$P_{\mathcal{P}/\mathcal{R}} + u_t V_{\mathcal{P}/\mathcal{W}} + u_t V_{sol\mathcal{W}/\mathcal{R}} \in T_1. \tag{11}$$

Infinity of solutions exists. We represent 3 of them in Figure 4, right, parameterized by α. $V_{sol}(\alpha = 0)$ corresponds to a pure orientation adaptation. $V_{sol}(\alpha = 1)$ corresponds to a pure velocity adaptation. Finally, $V_{sol}(\alpha = 0.5)$ corresponds to a combination of velocity and orientation adaptation (by orthogonal projection of I on T_1). We thus introduce a new parameter α which defines the avoidance strategy of the reference walker. We set the default value: $\alpha = 0.5$, this parameter is later calibrated from real data. Note that the personal area induces asymmetry in the model. When choosing the T_2 solution domain, (i.e., when giving way) more adaptation is required, and I reaches the limits of the interaction area later. It is even possible that no adaptation is required for the first walker passing, whilst the one giving way detects a need for adapting its motion. This property is correlated to our experimental observations.

4.2. Model Calibration

A walker's behavior is controlled by the five model parameters. α determines the walker's avoidance strategy while β_v, γ_v, β_θ and γ_θ control perception error evolution in time. We provide default values for each parameter above. We now propose to use the Maximum Likelihood Estimation (M.L.E.) technique [HS98] to calibrate our model from experimental samples. Briefly, this method consists of successive testing of different parameter sets. The one leading to the best match between real and simulated trajectories is identified. We initialize simulations from the measured experimental conditions (i.e., relative position and comfort velocities at the initiation of the interaction phase). More precisely, we search for the parameter set $p(\alpha, \beta_v, \gamma_v, \beta_\theta, \gamma_\theta)$ so that the likelihood estimator $\mathcal{L}(p)$ is maximum:

$$\hat{p} = \arg\max_p \mathcal{L}(p), \text{ where: } \mathcal{L}(p) = \prod_{i=0}^{n} f_p(\delta_i), \tag{12}$$



(a) Uncalibrated model (b) Calibrated model (c) RVO model (d) Reynolds' model (e) Helbing's model

Figure 5: *A direct comparison between real interaction (red trajectories) and simulated interaction (blue trajectories) for models.*

with δ_i the error between the experimental position Pe at time i and the simulated position Ps for an identical time (Manhattan distance is used):

$$\delta_i = \|\overrightarrow{PePs}\|_1, \tag{13}$$

$f_p(\delta)$ is the probability function of a normal distribution $\mathcal{N}(\mu,\sigma^2)$:

$$f_p(\delta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\delta-\mu)^2}{2\sigma^2}}. \tag{14}$$

Figure 5 illustrates the model calibration results. All plots show real trajectories (in red) superimposed with simulated ones (in blue). Simulated and real positions at identical times are linked by black line segments. The plot (a) is obtained from the uncalibrated version of our model, using the default parameter values. The plot (b) is obtained after calibration. In the latter case, perception errors are better estimated, allowing the model not to anticipate reaction too early. As a result, reaction is delayed and needs to be stronger: simulated adaptation more accurately corresponds to real data. In order to enable comparison to previous solutions, we also provide simulated data using van den Berg's model [vdBPS*08] (using the RVO library implementation), Reynold's steering behavior [Rey99] (using the OpenSteer implementation) and Helbing's model [HM95]. Plot (c) (RVO model) reveals the need of the asymmetrical personal area. Because of the circular representation, the correction of the walker passing second to its velocity is small. We also compare our results to the Reynolds' technique because the assumptions used in this solution meet several of our experimental observations: reactions are anticipated and collaborative. However, walkers have symmetric reactions, and anticipation time is near-constant (just lightly randomized). However, this value is relatively correct in the case of our initial conditions. We finally choose the Helbing's model because of the large interest showed in his approach, and the many existing variants. We further evaluate and validate our model in the following section, in comparison to these three approaches.

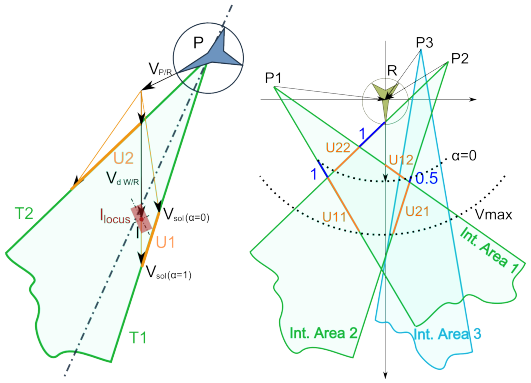


Figure 6: ***left:*** *Step 3 is modified to solve multiple interactions. The velocity domains U_1 and U_2 contain all possible solution velocities for $0 \leq \alpha \leq 1$.* ***right:*** *Step 4 and 5: the reference walker $\mathcal{R}$ needs to avoid a collision with perceived walkers $\mathcal{P}_1$ and $\mathcal{P}_2$. At the end of step 3, S contains 4 velocity domains: $S = \{U_{11}, U_{12}, U_{21}, U_{22}\}$ (orange color). After step 4, S' = { U'_{11}, U'_{21}, U'_{22}} (blue color), whilst U_{12} is dropped. The solution velocity, closest to the current one, V_{sol} belongs to U_{21} but due to the presence of $\mathcal{P}_3$, this solution is rejected and the final solution velocity V_{sol} belongs to U_{22}.*

4.3. Multiple Interactions

In the previous section, we describe and calibrate our interaction model from the pair-interaction case perspective. We are then able to handle sparsely populated environments where a walker is usually avoiding one or few other walkers at long distance without need for further computation. In this section, we describe how our model is able to solve simultaneous multiple interactions in more densely populated environments. Two supplementary steps - *step 4 and step 5* - are required to address such situations.

Similarly to the pair-interaction case, *steps 1* to *3* are processed for each of the n perceived walker $\mathcal{P}_i$, $i = 1,..,n$. Note that, for each pair-interaction, *steps 2* and *3* run only

	calibr. model	uncal. model	RVO model	Reynolds' steer.	Helbing's model
fig. 5 $\mathcal{L}(p)$	0.285	0.056	0,001	0.004	$1.3\ 10^{-15}$
mean $\mathcal{L}(p)$	0.214	0.165	0.14	0.047	0.004

Table 1: *values of the likelihood function using different simulation models. First results correspond to the example given Figure 5. Second results are mean values over 429 simulations corresponding to each of the available experimental sample.*

if the corresponding I_{locus} is inside the interaction area. Step 3 is however slightly modified. Instead of computing V_{sol} according to one specific α value, we compute two velocity domains $U_{i1} \in T_{i1}, U_{i2} \in T_{i2}$ respectively for $0 \leq \alpha \leq 1$. U_{i1} and U_{i2} are shown in Figure 6, left. Result is a set of solution velocity domains: $S = \{U_{i1}, U_{i2}\}$ with $i = 1,..,m$. Note that $m < n$ because some of the perceived walkers have no interaction with the considered reference walker. Real humans have a limited capability of considering several interactions simultaneously, we arbitrarily limit $m \leq 7$.

step 4 - Merging solution velocity domains. For each interacting perceived walker, i.e., for $i = 0$ to m, we compute the parts of the corresponding solution velocity domain U_{i1} and U_{i2} not intersecting the interaction areas corresponding to any other considered walker. We thus compute a merged solution velocity domain S' which contains reduced velocity domains U'_{i1} and U'_{i2}, computed as follows:

$$U'_{i1} = U_{i1} - U_{i1} \cup I_j, \text{ and } U'_{i2} = U_{i2} - U_{i2} \cup I_j, \qquad (15)$$

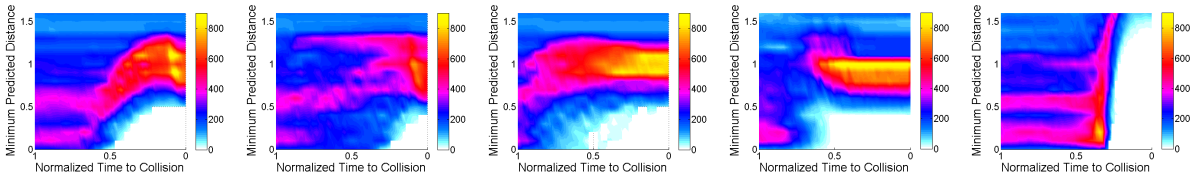
where I_j is the interaction area corresponding to the j^{th} perceived walker, $j = 0..m$, and $j \neq i$.

step 5 - Rating velocity domains. In case S' is empty, we set $V_{sol} = 0$. In the opposite case, each velocity belonging S' successfully solves all the considered interactions. However, they may lead to new interactions with nearby walkers, yet unconsidered. To diminish the number of interactions induced by the retained solution velocity, each possible solution is envisaged. We retain the solution velocity V_{sol} that minimizes the number of new interactions with walkers (i.e., not considered in the S interaction set). Steps 4 and 5 are illustrated in Figure 6, right. Note that $\mathcal{P}_3$ is not interacting with $\mathcal{R}$ at this precise simulation step, but makes $\mathcal{R}$ preferably avoiding $\mathcal{P}_1$ and $\mathcal{P}_2$ by turning to the right in order for him to not start interacting with $\mathcal{P}_3$.

5. Results

Quantitative model evaluation. The likelihood function $\mathcal{L}(p)$ can be directly used as a metric for a quantitative evaluation of our model results. Moreover, this function can also be evaluated for any simulated data which enables comparison between different approaches. Table 1 provides the obtained likelihood function value using: first, our calibrated model, second, our model using default parameter values, third, RVO model, fourth, Reynolds' steering behavior and finally, Helbing's model. The first line of results in Table 1 is computed from the example presented in Figure 5. The second line provides the mean value of the likelihood function computed over all the 429 available experimental samples. The higher the likelihood value, the more realistic the results. The obtained likelihood for the calibrated model is obviously higher than the one with default parameters value. Likelihood of the uncalibrated model is only slightly better than RVO model. However, this approach is not able to correctly simulate a large variety of cases, especially due to the near-constancy of anticipation times. Furthermore, adaptations are symmetrically made by the two walkers. Reynolds' model is worse than previous two and has the same disadvantages as the RVO model. In the case of Figure 5, the low contribution of the first walker passing is clearly observable. Only our solution correctly simulates such an asymmetry, even without calibration. Helbing's model is not adequate for this simulation setup. Finally, 398 times over 429 samples, the uncalibrated model correctly simulates the passage order between walkers. After calibration, the correct order is found 416 times. The obtained realism in our simulation results has no prohibitive computational cost: the three steps of one model iteration are averagely computed in $16\mu s$. (on a PC with Intel Core2-Duo X9000 at $2.8GHz$).

Qualitative model evaluation The qualitative comparison between models can be further detailed looking at the density plots displayed in Figure 7. Density plots enables a statistical evaluation of the relative duration of the observation, reaction and regulation periods, as well as the evolution of the *mpd* value. They are obtained from simulated data in exactly the same manner as for the one displayed in Figure 3 from experimental data. On plots (a) and (b) of Figure 7, the uncalibrated and calibrated model density plots are respectively shown. The uncalibrated version of the model over-increases the *mpd* value, the duration of the reaction period is however correctly simulated. The calibration delays reactions but the final distance between walkers at interaction time is now correctly regulated. The RVO's and Reynolds' model also converges toward a correct final distance between walkers. For RVO model, *mpd* is increasing smoothly but starts increasing from the moment the walkers can see each other. Reaction is apparently too abrupt concerning the Reynolds' method. This is due to simultaneous adaptations of walkers' motion; in reality, adaptations are not synchronized, which makes the reaction period longer and smoother. The lack of anticipation of Helbing's model is detectable, and the minimal distance between walkers overpass realistic values.



(a) Uncalibrated model (b) Calibrated model (c) RVO model (d) Reynolds' model (e) Helbing's model

Figure 7: *Trajectory density plots for 429 simulations, using initial conditions extracted from experimental samples, and using respectively (a) our model with default parameters values, (b) our model with calibrated parameters values, (c) the RVO model, (d) the Reynolds' model, and (e) the Helbing's model.*

6. Discussion

Our approach has several limitations. Doubtlessly, the major limitation is to have restricted our study on single pair-interactions. However, we discuss these limitations, as well as future work directions in the following paragraphs.

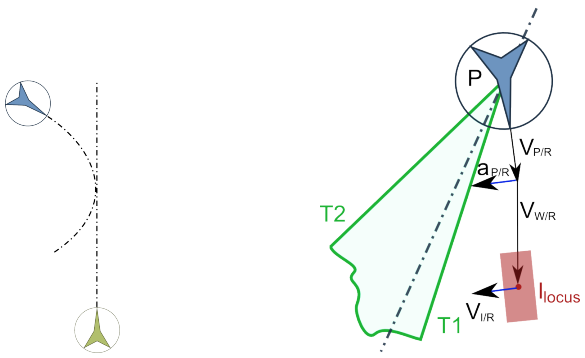


Figure 8: *A complex interaction situation where the perceived walker has non-linear trajectory.*

The non-linear case. So far, we have assumed that walkers have a constant velocity during the observation period of time. When a curved path is followed, such as in the example displayed in Figure 8, this assumption becomes false and the proposed model version is not able to correctly anticipate a collision avoidance. The problem comes from the position of the interaction point I which is moving relatively to the tangents T_1 and T_2 (by the interaction point definition, cf. Equation 7). When $\mathcal{P}$ accelerates, the instantaneous velocity of I in our representation is:

$$V_{I/\mathcal{R}} = u_t A_{\mathcal{P}/\mathcal{R}}. \quad (16)$$

In order to address such a situation, we adapt the step 1 of the model iteration. We compute two new variables t_{in} and t_{out}, which are respectively the time when I_{locus} is predicted to enter and next, leave, the interaction area. This evaluation is made from the instantaneous velocity of I, and distance from I_{locus} to T_1 and T_2 in the $V_{I/\mathcal{R}}$ direction. This estimation is coarse, given that I is probably not having a constant velocity in time. We then compare t_{in} and t_{out} to tti. If $tti \in [t_{in}, t_{out}]$, a collision is predicted. Then, two cases are considered. If tti is closer to t_{out} than t_{in}, motion adaptation is computed to make I_{locus} exit the interaction area more rapidly. Conversely, if tti is closer to t_{in}, motion adaptation is computed to avoid I_{locus} entering the interaction area. Steps 2 and 3 of the model iteration are modified accordingly. We illustrate such a complex situation in the companion video. As far as we know, none of the existing models is able to handle correctly such a case.

Orientation and velocity perception errors. We divide perception errors into two terms, ε_v and ε_θ. Our intuition is that the initial value of these two terms, when another walker is just perceived, depends on its relative position and walking direction. We assume that velocity is more accurately perceived than orientation when locomotion is perceived from a lateral point of view. Conversely, when walkers are face to face, orientation is more precisely perceived than velocity. In the proposed model, and in the example of the latter case, I_{locus} is elongated and rapidly contained into the interaction area due to its orientation. Consequently, this interaction is solved in simulations at far distances. This intuition is confirmed when thinking of interactions in straight corridors.

Taking obstacles into account. Environment obstacles have three major roles in interactions. First, they limit walkers' visual perception. Obviously, an interaction is initiated between two walkers when they are able to see each other. Second, they may limit the solution velocity domain. As for multiple interactions, it is possible to solve interactions with such limitations. Singh and colleagues recently proposed a framework to evaluate walker-walker interaction solving among obstacles [SNK*08]. Accounting for obstacles in a general manner, and evaluating our results using the proposed framework is in progress. We provide first results in the companion video: one walker cannot deviate during the interaction due to the presence of an obstacle. Finally, areas made invisible by obstacles have a role in locomotion. For instance, at corridors crossing, walkers adapt their motion because they expect someone to appear from an invisible area.

7. Conclusion

We presented a novel approach to simulate interactions between virtual walkers. The model is based on a detailed experimental study, allowing us to observe how real humans behave in such situations. We state that the *minimum predicted distance* is an adequate criterion to determine whether humans require to adapt their trajectory or not. We also demonstrated that humans react after an observation period during which perceived motion is estimated more and more accurately. Interactions are solved by a combination of velocity and orientation adaptations, which is role-dependent. The task is collaborative between the two walkers, however, the walker passing in front of the one giving way makes quantitatively less adaptations. We proposed a model able to simulate and reproduce our experimental trajectories. Our model has few parameters and can be automatically calibrated on real data. We evaluated our approach and compared it to previous techniques. We demonstrated the achieved improvements: our model is able to determine correctly if, when and how motion is adapted to solve interactions. Future work's main objective is to study multiple and complex interactions between walkers among obstacles, and to extend our model accordingly. We will then be able to validate our main hypothesis, which states that a complex interaction can be decomposed into a combination of pair-interactions. Understanding how humans combine such pair-interactions is then crucial.

References

[ALHB08] ARECHAVALETA G., LAUMOND J.-P., HICHEUR H., BERTHOZ A.: On the nonholonomic nature of human locomotion. *Auton. Robots 25*, (1-2) (2008), 25–35.

[BJ03] BROGAN D. C., JOHNSON N. L.: Realistic human walking paths. In *CASA '03: Proc. of the 16th Int. Conference on Computer Animation and Social Agents (CASA 2003)* (Washington, DC, USA, 2003), IEEE Computer Society, p. 94.

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

[Feu00] FEURTEY F.: *Simulating the Collision Avoidance Behavior of Pedestrians*. PhD thesis, Department of Electronic Engineering, Master's thesis, University of Tokyo., 2000.

[GLRM05] GÉRIN-LAJOIE M., RICHARDS C., MCFADYEN B.: The negociation of stationary and moving obstructions during walking : anticipatory locomotor adaptations and preservation of personal space. *Motor Control 9* (2005), 242–269.

[Gof71] GOFFMAN E.: *Relations in public : microstudies of the public order*. New York : Basic books, 1971.

[HBJW05] HELBING D., BUZNA L., JOHANSSON A., WERNER T.: Self-organized pedestrian crowd dynamics: experiments, simulations and design solutions. *Transportation science 39*, (1) (2005), 1–24.

[HLTC03] HEIGEAS L., LUCIANI A., THOLLOT J., CASTAGNÉ N.: A physically-based particle model of emergent crowd behaviors. In *Graphicon 2003* (2003).

[HM95] HELBING D., MOLNAR P.: Social force model for pedestrian dynamics. *Physical Review E 51* (1995), 4282.

[HS98] HARRIS J. W., STOCKER H.: *Handbook of Mathematics and Computational Science*. Springer-Verlag, N-Y., 1998, ch. Maximum Likelihood Method, p. p. 824.

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

[LCHL07] LEE K. H., CHOI M. G., HONG Q., LEE J.: Group behavior from video: a data-driven approach to crowd simulation. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 109–118.

[LCL07] LERNER A., CHRYSANTHOU Y., LISCHINSKI D.: Crowds by example. *Computer Graphics Forum (Proceedings of Eurographics'07) 26*, (3) (2007).

[Lee76] LEE D.: A theory of visual control of braking based on information about time-to-collision. *Perception 5*, (4) (1976), 437–459.

[MH04] METOYER R. A., HODGINS J. K.: Reactive pedestrian path following from examples. *The Visual Computer 20*, (10) (2004), 635–649.

[PAB07] PELECHANO N., ALLBECK J. M., BADLER N. I.: Controlling individual agents in high-density crowd simulation. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 99–108.

[PPD07] PARIS S., PETTRÉ J., DONIKIAN S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum : Eurographics'07 26*, (3) (2007), 665–674.

[Rey99] REYNOLDS C. W.: Steering behaviors for autonomous characters. In *Game Developers Conference 1999* (1999).

[SNK*08] SINGH S., NAIK M., KAPADIA M., FALOUTSOS P., REINMAN G.: Watch out! a framework for evaluating steering behaviors. In *MIG* (2008), Egges A., Kamphuis A., Overmars M. H., (Eds.), vol. 5277 of *Lecture Notes in Computer Science*, Springer, pp. 200–209.

[TCP06] TREUILLE A., COOPER S., POPOVIĆ Z.: Continuum crowds. *ACM Transactions on Graphics (SIGGRAPH 2006) 25*, (3) (2006).

[TLP07] TREUILLE A., LEE Y., POPOVIĆ Z.: Near-optimal character animation with continuous control. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, (3) (2007).

[Tré91] TRÉSILIAN J.: Empirical and theoretical issues in the perception of time to contact. *Journal of Experimental Psychology Human Perception and Performance 17 (3)* (1991), 865–876.

[vdBPS*08] VAN DEN BERG J., PATIL S., SEWALL J., MANOCHA D., LIN M.: Interactive navigation of individual agents in crowded environments. *Symposium on Interactive 3D Graphics and Games (I3D 2008)* (2008).

A Synthetic-Vision Based Steering Approach for Crowd Simulation

Jan Ondřej* INRIA
Julien Pettré* INRIA
Anne-Hélène Olivier* INRIA
Stéphane Donikian* INRIA
Golaem S.A.

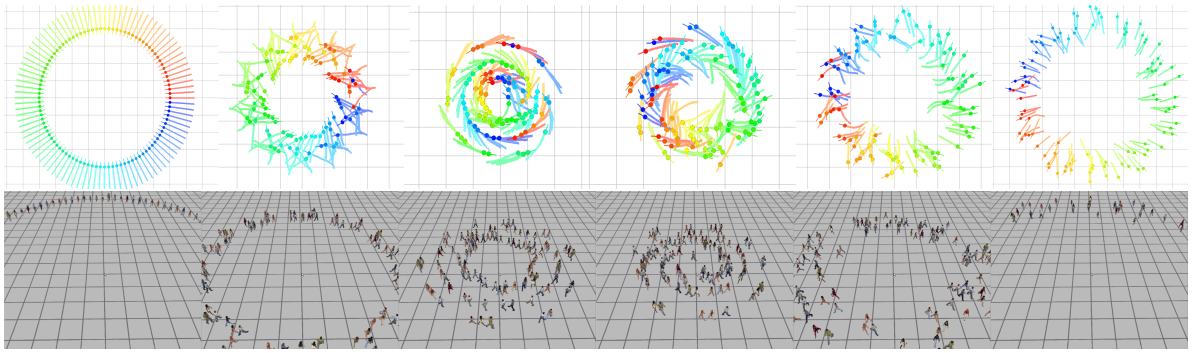


Figure 1: *Animations resulting from our simulations. Emergent self-organized patterns appear in real crowds of walkers. Our simulations display similar effects by proposing an optic flow-based approach for steering walkers inspired by cognitive science works on the human locomotion. Compared to previous approaches, our model improves such an emergence as well as the global efficiency of walkers traffic. We thus enhance the overall believability of animations by avoiding improbable locking situations.*

Abstract

In the everyday exercise of controlling their locomotion, humans rely on their optic flow of the perceived environment to achieve collision-free navigation. In crowds, in spite of the complexity of the environment made of numerous obstacles, humans demonstrate remarkable capacities in avoiding collisions. Cognitive science work on the human locomotion stated that a relatively succinct information is extracted from the optic flow to achieve a safe locomotion. In this paper, we explore a novel vision-based approach of collision avoidance between walkers that fit the requirements of interactive crowd simulation. In imitation of humans and based on cognitive science results, we detect future collisions as well as their dangerousness from visual-stimuli. The motor-response is twofold: reorientation strategy is set to avoid future collision, whereas a deceleration strategy is used to avoid imminent collisions. Several examples of our simulation results show that the emergence of self-organized patterns of walkers is reinforced using our approach. Emergent phenomena are visually appealing. More importantly, they improve the overall efficiency of the walkers traffic and allow avoiding improbable locking situations.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.6.5 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: crowd simulation, steering method, collision avoidance, synthetic vision

*e-mail:{jan.ondrej,julien.pettre,anne-helene.olivier,donikian}@irisa.fr

1 Introduction

Crowd simulation has significantly grown in importance these two past decades. Their field of application is wide and ranges from the domains of security and architecture to the one of movie industry and interactive entertainment. The visually impressive self-organized patterns that emerge at a large-scale from the combination of all the local actions and interactions in crowds is probably a major reason of the attention paid by Computer Animation on this topic. Reynolds' seminal work on flocks of *boids* showed that fascinating global motions can be obtained from simple local interactions rules [Reynolds 1987]; however, the proposed rules explicitly stick boids together to obtain emerging flocks. Moreover, boids motion rules are not directly transposable to human walkers.

Human crowds are the place of numerous and various interactions. In this paper, we focus on crowds of *individually walking humans* where interactions are limited to *collision avoidance*. Our motivation is to design a local collision avoidance method that remains as close as possible to the real human behavior while displaying emerging self-organized patterns as witnessed in real crowds. This objective is representative of our bottom-up approach: specific large-scale formations are expected from realistic local interactions between walkers. Simulating emerging formations is crucial in order to obtain believable crowd animations. Obtaining them from individually steered walkers avoiding each-other, and thus, simulating self-organization, is particularly challenging.

Collision avoidance has recently received much attention. Several types of approach were proposed (cf. Section 2 for an overview). Most of recent *agent-based* techniques are based on geometrical models. Their common point is to explicitly compute admissible velocities that allow avoiding future collisions: efforts are focused on reaching highest performance in order to handle large crowds. Then, challenge is to steer walkers with believable trajectories while remaining in the admissible velocity domain. However, geometrical models are also disconnected from reality since humans unconsciously react to perceived obstacles to avoid collisions. This raises fundamental question, that is can simpler perception/action control loops - probably closer to reality - steer virtual walkers and allow them avoiding collisions even in complex situations? *Rule-based* techniques explored such a question; however, artifacts occur in most complex situations because of the difficulty in combining rules. *Particle-systems* and *continuum-based* methods ease the combination of interactions and are able to handle even larger crowds. They however have drawbacks as well. The former sometimes fail to simulate emerging patterns of walkers while the latter may lead to unrealistic local motions as for example unfeasible accelerations or velocities.

In contrast with previous approaches, we steer walkers according to the visual perception they have of their environment. We thus formulate our collision avoidance solution as a visual-stimuli/motor-response control law. Our model is inspired by the work of Cutting and colleagues [1995] on the human locomotion in the field of cognitive science. They stated that humans extract two major elements from their optic flow to achieve collision-free navigation. First is the derivative of the bearing-angle under which obstacles are perceived. Second is the time-to-collision which is deduced from the rate of growth of obstacles in successively perceived images. Inspired by these observations, our model's inputs, i.e., the visual-stimuli, are the egocentrically perceived obstacles transformed into images of time-derivatives of bearing-angles and of times-to-collision. These images are directly computed from the geometries and states of both the static and moving obstacles of the scene. Walkers have simple reactions to these stimuli: they turn to avoid future collisions and decelerate in the case of imminent collisions.

Our contributions are thus the following. We propose a vision-based collision avoidance model for interactive simulation of crowds of individual humans. We base our approach on cognitive science work on the human locomotion, which inspired us novel local visual-stimuli/motor-response laws. We apply our method to complex situations of interaction: resulting simulations display the emergence of interesting self-organized patterns of walkers at a global-scale. We demonstrate our improvements in comparison to previous approaches, with enhanced emergence of patterns of walkers, improved global efficiency of the walkers traffic, and smoother animations.

The remainder of the paper is organized as follows. Section 2 first provides an overview of crowd simulation techniques with particular focus on collision avoidance methods. Following, we present the guiding principles of our approach before describing the proposed model in details in Section 3. We provide details about its implementation in Section 4. Finally, we illustrate simulation results from several examples and give comparison with previous techniques in Section 5. Limitations of our approach and future work are discussed in Section 6, before concluding.

2 Related Work

Virtual Crowd is a wide topic that raises numerous problems including population design, control, simulation or rendering and was surveyed in recent books [Thalmann and Raupp Musse 2007; Pelechano et al. 2008] and tutorials [Thalmann et al. 2005; Halperin et al. 2009]. This overview focuses on crowd simulation, the objective of which can be restrictively defined as computing global locomotion trajectories to achieve goal-driven collision-free navigation for crowds of walkers.

Several classes of solutions were proposed in the literature. Cellular-automaton approaches [Schadschneider 2001] are used to simulate evacuation scenarios for large crowds: the discrete aspect of resulting trajectories prevent their use for Computer Animation applications. However, grid-based solutions were adapted to meet such requirements [Loscos et al. 2003], and for example Shao and Terzopoulos [2005] proposed the use of multi-resolution grids to handle large environments. Other techniques consider velocity-fields to guide crowds [Chenney 2004]. This analogy with Physics gave rise to particle-systems approaches. Helbing [1995] proposed the social-forces model where walkers repulse each-other while they are attracted by their goal. The social forces model was later revisited in [Pelechano et al. 2007; Gayle et al. 2009]. Evolved models proposed using mass-damp-spring systems to compute similar repulsing forces between walkers [Heigeas et al. 2003]. Crowd simulation was also studied as a flowing continuum [Hughes 2003; Treuille et al. 2006] that allows simulating numerous walkers in real-time. Even larger crowds were handled using hybrid continuum-based approach [Narain et al. 2009]. From a general point-of-view, high computation performance is a common point between all of these approaches. Such performance allows simulating large crowds in equally large environments in real-time, which is a crucial need of many interactive applications. Performance is however obtained at the cost of some limitations, such as restricting the total number of goals walkers can have, or using of simplistic interaction models that may lower the realism of results. Compared to this former set of approaches, our first objective is not to reach a high-performance solution but to simulate local interactions in a realistic manner. By realism, we here mean that we reproduce the human vision-based locomotion control in order to steer walkers in crowds. Synthetic vision raises numerous computations by nature.

Our method can be closely related to rule-based approaches [Reynolds 1999] as well as to geometrically-based local avoidance models approaches [Paris et al. 2007; van den Berg et al. 2008; Kapadia et al. 2009; Pettré et al. 2009; Karamouzas et al. 2009; Guy et al. 2009]. It is generally required to combine local approaches with dedicated techniques in order to enable the reaching of high-level goals in complex environments [Lamarche and Donikian 2004; Paris et al. 2006; Pettré et al. 2006; Sud et al. 2007]. Nevertheless, geometrically-based avoidance models carefully check the absence of future collisions locally, given the simulation state. This goal is generally achieved by decomposing the reachable velocity-space of each walker into two components: the inadmissible velocity domain and the admissible velocity domain. These domains respectively correspond to velocities leading to collisions and those allowing avoidance. At the opposite, our method make walkers react to some situations without explicitly computing the admissibility of their motion adaptations. This raises a fundamental question: can explicit collision checks guarantee the absence of residual collisions? We argue the answer is negative. The reason is twofold. First, the admissible velocity domain is computed assuming that the velocity of moving obstacles remains constant. Second, the admissible velocity domain is often made of several independent components, especially in the case of complex interactions - i.e., during simultaneous interactions with several obstacles. Some of these components degenerate in time because moving obstacles may also adapt their own motion. If the current velocity of a given walker belong such a degenerative component, switching to another component is required. As a

result, traversing the inadmissible velocity domain is required when acceleration is bounded, whereas unbounded accelerations result into unrealistic motions. Our method do not explicitly check collisions and is not exempt from failure. We however believe the proposed visual-stimuli/motor-response laws better imitate the most basic level of real human locomotion control.

We previously addressed the question of realism of simulated locomotion trajectories during collision avoidance in [Pettré et al. 2009]. We provide a qualitative description of such trajectories: we experimentally show that real humans anticipate avoidance as no more adaptation is required some seconds before walkers pass at close distance. We also show that avoidance is a role-dependent behavior as the walker passing first makes noticeably less adaptations than the one giving way. We discuss the visual information humans may exploit to be able to achieve avoidance in such a manner. However, we proposed a *geometrical model* to reproduce such trajectories that is calibrated from our experimental dataset. Compared to this work, we here address two new problems. First, we address the question of combining interactions. We explore synthetic vision as a solution to implicitly combine them, for example: they are integrated by projection to the perception image, they are filtered when obstacles are invisible, they are weighted by the importance obstacles have in the image. Second, we directly base our motion control laws on the visual information believed to be exploited by real humans.

Vision-based methods were never used to tackle the crowd simulation problem to the best of our knowledge, with the exception of Massive software agents [Massive] which are provided with synthetic vision; however, controlling walkers from such an input is left at the charge of users. Nevertheless, synthetic vision was used to steer a single or few virtual humans [Noser et al. 1995; Kuffner and Latombe 1999; Peters and O'Sullivan 2003] or artificial creatures [Tu and Terzopoulos 1994]. Reynolds' boids were also recently provided with visual perception abilities [Silva et al. 2009]. Our approach explores a new type of visual-stimuli to control locomotion, based on statements from cognitive science. We also improve performance to fit the requirements of interactive crowd simulation. Finally, visual-servoing is an active topic in the field of Robotics [Chaumette and Hutchinson 2006]. Major challenges are processing optic flows acquired with physical systems and extracting the relevant information that allow steering robots. In contrast to this field, we do not process digitally computed images but directly compute the required visual-inputs of our model.

3 Vision-based collision avoidance

3.1 Model overview

Humans control their locomotion from their vision [Warren and Fajen 2004]. According to Cutting and colleagues [Cutting et al. 1995] humans successively answer two questions during interactions with static and moving obstacles: will a collision occur? When will collision occur? Cutting experimentally observed that these two questions are answered by extracting two indicators from the perceived optic flow:

1. *Will a collision occur?* Humans visually perceive obstacles under a given angle referred to as the *bearing-angle* (noted α). A collision is predicted when the time derivative of the bearing angle, $\dot{\alpha}$, is zero (or close to zero because of the body envelopes). This observation is illustrated in Figure 2 from the 3 examples of two walkers displaying converging trajectories.

2. *When will collision occur?* Humans visually perceive obstacles with given sizes. The rate-of-growth of obstacles in time allow humans to detect obstacles coming toward them when positive. Moreover, the higher the rate the more imminent the collision. As a result, humans are able to evaluate the time-to-collision (ttc).

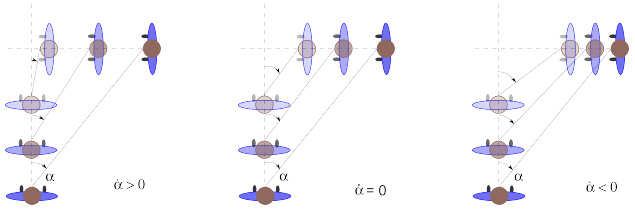


Figure 2: *The bearing-angle and its time-derivative, respectively α and $\dot{\alpha}$, allow detecting future collisions. From the perspective of an observer (the walker at the bottom), a collision is predicted when α remains constant in time.* ***(left)*** *$\alpha < 0$ and $\dot{\alpha} > 0$: the two walkers will not collide and observer will give way.* ***(center)*** *the bearing-angle is constant ($\dot{\alpha} = 0$). The two walkers will collide.* ***(right)*** *$\alpha < 0$ and $\dot{\alpha} < 0$: the two walkers will not collide and observer will pass first.*

Therefore, the relevant information necessary to achieve collision-free locomotion according to Cutting is entirely described by the pair $(\dot{\alpha}, ttc)$. It is to notice that humans use similar information to intercept mobile targets as described by Tresilian in [Tresilian 1994].

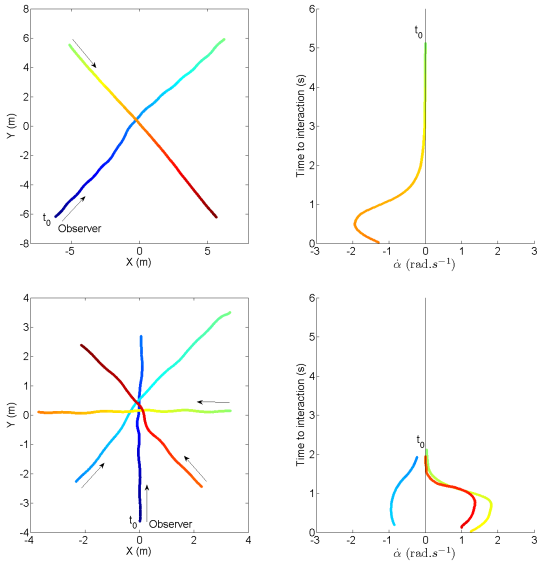


Figure 3: *Two examples of real interactions between (top) two walkers and (bottom) four walkers. Motion captured trajectories projected on the ground are shown (plots on the left), as well as in the $(\dot{\alpha}, tti)$-space (plots on the right), as perceived by one of the participant called 'observer'. Trajectories are colored in order to enable matching between the two representations.*

Figure 3 illustrates Cutting's theory from 2 examples of real interactions: trajectories are displayed in the horizontal plane as well as in the $(\dot{\alpha}, tti)$-space, where tti is the time-to-interaction. Time-to-interaction is the time remaining before minimum distance between participants is reached, according to current positions and velocities. The notion of time-to-collision ttc is generally used in the literature in place of our time-to-interaction tti; these two notions are

close. By definition ttc exists if and only if a risk of future collision is also existing. At the opposite, tti exists whatever the relative positions and velocities of the considered moving objects. Also note that tti can reach negative values when the considered objects display diverging motions. In the first example, we observe that $\dot{\alpha}$ is initially close to zero whilst tti decreases: collision is predicted. By turning to the left, the observer solves the interaction: $\dot{\alpha}$ decreases. On the second example, future collision with the observer is predicted for two walkers among the three perceived ones. By turning and decelerating, $\dot{\alpha}$ values are corrected. The impact of motion adaptations on the variations of $(\dot{\alpha}, tti)$ is not intuitive. However, as a first approximation, turns mainly plays on the $\dot{\alpha}$ value, whereas a deceleration mainly changes tti.

The guiding principles of the proposed model - based on Cutting's results - are thus the following. A walker perceives the static and moving obstacles of his environment as a set of points $P = \{p_i\}$ resulting from his synthetic vision. For each perceived point p_i, we compute the bearing angle α_i, its time-derivative $\dot{\alpha}_i$, and the remaining time-to-interaction relatively to the walker tti_i. We deduce the risk of a future collision from $\dot{\alpha}_i$. We also deduce the dangerousness of the situation from tti_i. A walker reacts when needed according to two strategies. First, he avoids future collision by adapting his orientation with anticipation. Second, in the case of an imminent collision, he decelerates until he gets stopped or the interaction is solved. The following sections detail how we put these principles into practice.

3.2 Model inputs

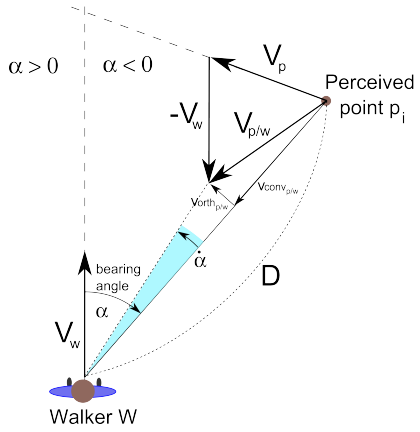


Figure 4: *Model's inputs. Any point is perceived under given bearing-angle. The triad $(\alpha_i, \dot{\alpha}_i, tti_i)$ is deduced from the relative point position and velocity with respect to the walker.*

A walker configuration is defined by its position and orientation θ. He is velocity-controlled by his angular velocity $\dot{\theta}$ and his tangential velocity v. Perceived points $p_i \in P$ may indiscriminately belong to static obstacles - such as walls - or moving ones - such as other walkers. Also note that a single obstacle result in several points with respect to its shape: Figure 6 illustrates how a walker perceives his environment. The variables associated to each $p_i \rightarrow (\alpha_i, \dot{\alpha}_i, tti_i)$ are deduced from the relative position and velocity of p_i to the walker; we however detail their computation in Figure 4 as well as in the following Implementation Section 4.

3.3 Angular velocity control

As explained in the previous section, a walker detects a risk of future collision when $\dot{\alpha}$ is *low* and $tti_i > 0$. We define the $\dot{\alpha}_i$ threshold τ_1 under which a walker reacts as a function of the perceived tti_i as follows:

$$\tau_1(tti) = \begin{cases} \tau_{1-}(tti) = a - b.tti^{-c} & \text{if } \dot{\alpha}_i < 0, \\ \tau_{1+}(tti) = a + b.tti^{-c} & \text{otherwise.} \end{cases} \quad (1)$$

where a, b and c are some parameters of the model. These three parameters change a walker avoidance behavior by adapting his anticipation time as well as the security distance he maintains with obstacles. We detail the role of these parameters in the Discussion Section 6. Figure 5 plots the function τ_1 for $a = 0$, $b = 0.6$ and $c = 1.5$. These values were used in the examples shown in Section 5, and were determined by manually fitting τ_1 on numerous experimental data capturing avoidance between real walkers similar to those shown in Figure 3. Then, the set P_{col} of points $p_i(\dot{\alpha}_i, tti_i)$ a walker has to react to is defined as follows:

$$p_i \in P_{col} \text{ if } tti_i > 0 \text{ and } \alpha_i < \tau_1(tti_i) \quad (2)$$

We now combine the influence of the set of points belonging to P_{col}. For this purpose, we decompose P_{col} into P_+ and P_-, which respectively correspond to points with positive and negative $\dot{\alpha}_i$ values. We then define ϕ_+ and ϕ_- as follows:

$$\phi_+ = min(\dot{\alpha}_i - \tau_{1+}(tti_i)), \quad \text{for all } p_i \in P_+ \quad (3)$$
$$\phi_- = max(\dot{\alpha}_j - \tau_{1-}(tti_j)), \quad \text{for all } p_j \in P_- \quad (4)$$

At this point, we have identified all interactions requiring walkers to turn to the right to avoid future collision into P_+, and those asking to turn left into P_-. The required amplitude of a right turn allowing to avoid at once all the interactions provoked by the P_+ set of points directly depends on the amplitude of ϕ_+ (the same for a left turn, P_- and ϕ_- respectively). However, we must ensure walkers do not highly deviate from their goal. For this reason, we now consider the bearing-angle corresponding to the goal α_g, as well as its time-derivative $\dot{\alpha}_g$. Contrarily to obstacles, walkers attempt to intercept their goal, which means that $\dot{\alpha}_g = 0$ is desired. Three cases are then successively considered. Firstly, when $\dot{\alpha}_g$ is small (we arbitrarily choose $|\dot{\alpha}_g| < 0.1 rad.s^{-1}$), walkers are currently heading to their goal, the influence of which is neglected. In this case, we simply choose the change of direction which asks the minimum deviation. $\dot{\theta}$ is controlled as follows:

$$\dot{\theta} = \begin{cases} \phi_+ & \text{if } |\phi_+| < |\phi_-|, \\ \phi_- & \text{otherwise.} \end{cases} \quad (5)$$

Secondly, when $\phi_- < \dot{\alpha}_g < \phi_+$, but cannot be neglected, we choose the change of direction that leads to the smallest deviation from the goal. Then,

$$\dot{\theta} = \begin{cases} \phi_+ & \text{if } |\phi_+ - \dot{\alpha}_g| < |\phi_- - \dot{\alpha}_g|, \\ \phi_- & \text{otherwise.} \end{cases} \quad (6)$$

Thirdly, when $\dot{\alpha}_g < \phi_-$ or $\dot{\alpha}_g > \phi_+$ we choose:

$$\dot{\theta} = \dot{\alpha}_g \quad (7)$$

To avoid unrealistic angular velocities, $\dot{\theta}$ and $\ddot{\theta}$ are finally bounded so that $|\dot{\theta}| < \pi/2 (rad.s^{-1})$ and $|\ddot{\theta}| < \pi/2 (rad.s^{-2})$.

3.4 Tangential velocity control

Tangential velocity v is set to comfort velocity v_{comf} by default. It is only adapted in the case of a risk of imminent collision. The imminence of a collision is detected when tti_i is positive but lower

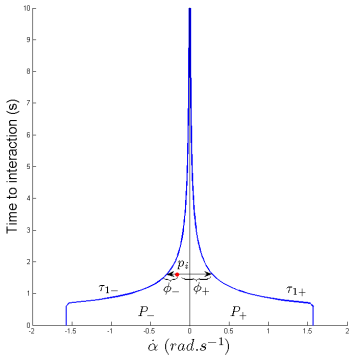


Figure 5: *τ_1 plot using the following parameter set: $a = 0$, $b = 0.6$ and $c = 1.5$ (cf. Equation (1)). Future collision is detected when $p_i(\dot{\alpha}_i, tti_i)$ is below τ_1 and $tti_i > 0$. The plot also illustrates that the lower the tti_i value, the higher the walker's reaction.*

than a threshold τ_2 (we arbitrarily choose $\tau_2 = 3s$.). Tangential velocity is controlled from the minimum positive tti_{mp} value perceived by the walker. We define P_{pos} the set of points $p_i \in P_{col}$ for which $tti_i < \tau_2$ and compute tti_{mp} as follows:

$$tti_{mp} = min(tti_i) \text{ for all } p_i \in P_{pos} \tag{8}$$

Finally, the walker's tangential velocity is controlled as follows:

$$v = \begin{cases} v_{comf} & \text{if } P_{pos} = \emptyset, \\ v_{comf}.(1 - e^{-0.5tti_{mp}^2}) & \text{otherwise.} \end{cases} \tag{9}$$

Position and orientation of the walker are finally updated according to the computed v and $\dot{\theta}$ values, with bounded $\dot{v}$ ($|\dot{v}| < 1m.s^{-2}$).

4 Implementation

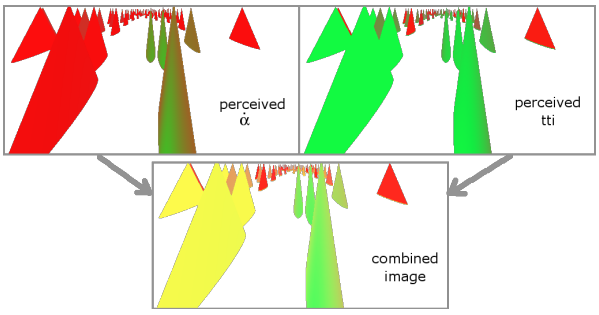


Figure 6: *Walkers perceive the environment obstacles as a set of points $p_i(\dot{\alpha}_i, tti_i)$. The image corresponding to all the perceived $\dot{\alpha}_i$ values of p_i is shown top-left (red are for lowest values). The image corresponding to all the perceived tti_i values of p_i is shown top-right (red are for lowest values). Perception is combined (bottom image) to compute walker reaction. In this example - which corresponds to the circle example, cf. Section 5 - the walker will react to the most red points of the combined perception. In this particular situation, he is likely to follow the walker in front of him on his right.*

We implemented our model using OpenGL, shader programming language and CUDA. The algorithm is decomposed into two major stages. First, for each virtual walker:

Step 1 Set camera position and orientation at the one of the considered walker (see details below).

Step 2 Render to texture environment obstacles using simplified geometries. Compute values α_i, $\dot{\alpha}_i$ and distance to obstacle d per vertex (Figures 4 and 6).

Step 3 Then, using a fragment shader, compute per pixel tti_i, build P_+ and P_- from τ_{1+} and τ_{1-}.

Step 4 Copy the resulting texture to the CUDA space and make a parallel reduction to compute ϕ_+, ϕ_-. Result is stored to an array on the GPU.

At the end of this first loop, the resulting array is downloaded once to the CPU. Then, for each walker again:

Step 5 Compute $\dot{\alpha}_g$ and deduce $\dot{\theta}$ and v.

Step 6 Update walker's position accordingly.

Camera Setup Walkers visually perceive their environment through the OpenGL camera set at the first step of our algorithm. The camera field-of-view is $150°$ of width and $80°$ of height. The camera position is set at the one of the considered walker at his eye level, and panning angle is aligned on the walker's motion direction. Tilting angle is set so that the upper clipping plane is horizontal (i.e., camera is oriented toward the ground with a $-40°$ angle). Resolution is 256×48 pixels.

Simplified Geometries The complexity of the proposed algorithm is dependent on the one of the environment (Step 2). Walker do not need to react to subtle geometrical details of the scene. Simplified bounding geometries can be used for obstacles. In particular, perceived walkers are geometrically simplified as cones of $1.8m$ of height and $0.5m$ of base radius. Cones, similarly to walking humans, are wider at their base than at their top. Real humans can see above others' shoulders: cones better reflect this ability than cylinders, for instance.

Computation of Inputs Model's inputs are computed as illustrated in Figure 4. In the figure, p_i is one of the perceived points that belongs to a given obstacle o. The relative velocity $\vec{V}_{p_i/w}$ of a perceived point with respect to the considered walker are first deduced:

$$\vec{V}_{p_i/w} = \vec{V}_o - \vec{V}_w \tag{10}$$

where $\vec{V}_w$ the walker's velocity vector and $\vec{V}_o$ the obstacle's velocity vector the perceived point belongs to. Finally, $\vec{V}_{p_i/w}$ is decomposed into $\vec{V}_{conv_{p_i/w}}$ and $\vec{V}_{orth_{p_i/w}}$ to deduce tti_i and $\dot{\alpha}_i$ ($\vec{V}_{conv}$ is for the component of the relative velocity converging to the considered walker, and $\vec{V}_{orth}$ the orthogonal one):

$$\vec{V}_{conv_{p_i/w}} = (\vec{V}_{p_i/w}.\vec{k}).\vec{k} \tag{11}$$

$$\vec{V}_{orth_{p_i/w}} = \vec{V}_{p_i/w} - \vec{V}_{conv_{p_i/w}} \tag{12}$$

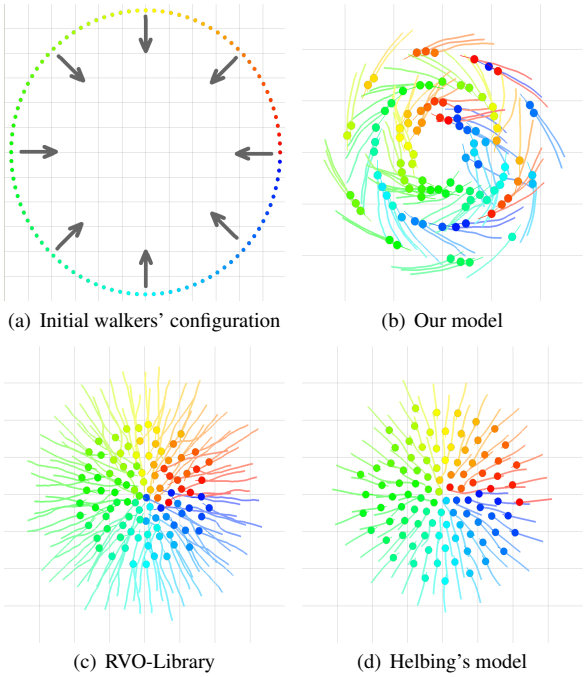
$$tti_i = D.\|\vec{V}_{conv_{p_i/w}}\|^{-1} \tag{13}$$

$$\dot{\alpha}_i = \arctan\left(\frac{\|\vec{V}_{orth_{p_i/w}}\|}{D - \|\vec{V}_{conv_{p_i/w}}\|}\right).u^{-1} \tag{14}$$

where D is the p_i-walker distance, $\vec{k}$ is the unitary p_i-walker vector, and u the unit of time.

5 Results

5.1 Examples



(a) Initial walkers' configuration (b) Our model

(c) RVO-Library (d) Helbing's model

Figure 7: ***Circle*** *(a) A scene of 100 walkers are initially deployed uniformly along a circle. Walkers goal is to reach the diametrically opposed position. Solution is shown for 3 models (b), (c), (d). Our model (b) is the only one able to provoke the emergence of patterns.*

We illustrate our simulation results according to four examples. Comparison with two previously existing techniques is provided for the two first examples in order to illustrate the achieved improvements. We chose:

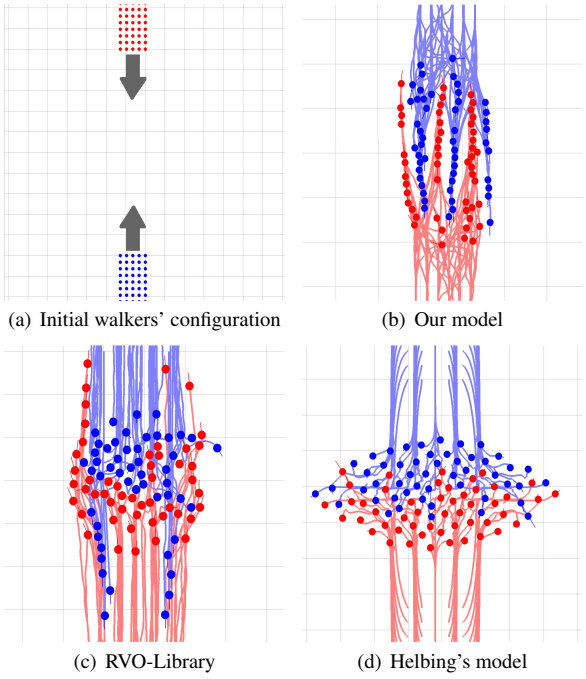
RVO which is representative of geometrical avoidance models.

Helbing's model which is representative of particle-based approaches. Contrarily to RVO, such models do not take into account anticipation and interactions are formulated as function of distance to obstacles.

The available examples are:

Circle: In this example, walkers are initially located along a circle and each one's goal is to reach the diametrically opposed position. In absence of others, each walker would go through the circle passing by its center. The number of interactions occurring in such an example is thus maximized: actually, each walker interacts with all the other ones. The main difficulty raised by this example is avoiding that walkers immediately converge to the center of the circle and get stuck there. Such situation can be efficiently avoided when 'traffic circles' emerge, whilst the center is almost left empty of anyone. Results are shown in Figure 1 and 7.

Group-swap: In this example, walkers are initially separated in two groups. The goal is to swap group positions. Motion is not constrained by static obstacles. A main difficulty raised by this example is to achieve collision avoidance whilst walkers do not excessively deviate from the shortest route in spite of the absence of constraints (e.g., corridor walls). Such a result can be reached only if lane formations emerge. Results are shown in Figure 8.



(a) Initial walkers' configuration (b) Our model

(c) RVO-Library (d) Helbing's model

Figure 8: ***Group-swap*** *A scene with two groups of walkers heading toward each other solved by three different models. In our model (b) distinct lane formations emerge with anticipation. The lane formations in RVO-Library (c) start emerging lately and lead to a congestion. Helbing's model (d) no such formation emerge.*

Pillars: In this example, we increase the difficulty of the group-swap example by adding two rows of pillars in the middle of the scene. We also demonstrate the ability of our model to take in account static obstacles. Results are shown in Figure 9.

Crossing: In this example, two groups of people meet at the intersection of two orthogonal corridors: static obstacles both constraints the motion and prevent walkers to early perceive the ones from the other group. Main difficulties of this example are: first, avoid that one of the two groups get stuck and second, avoid walkers to be excessively deviated along corridors walls. Results are shown in Figure 10.

All of the displayed examples demonstrate our model ability to let

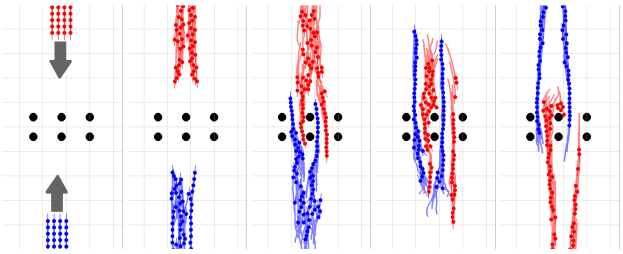


Figure 9: ***Pillars*** *This example is identical to the group-swap one, two rows of pillars make the scene more complex. The images show the evolution in time of the simulation starting from the left.*

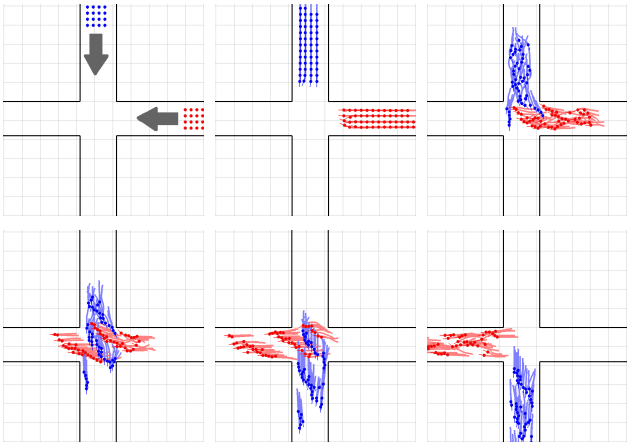


Figure 10: ***Crossing*** *This example shows two groups meeting at the intersection of orthogonal corridors. The emerging line patterns the direction of which is approximately* $45°$ *allows efficient global motion (evolution in time is shown from top-left to bottom-right).*

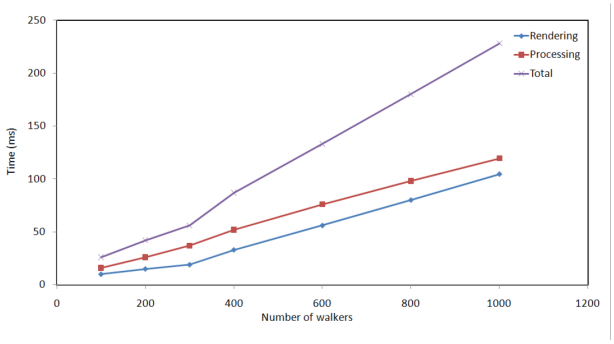


Figure 11: *Performance plot: computation time for one simulation loop with respect to the number of walkers is measured. Simulation ran on a laptop with Intel T7800@2.6GHz CPU and Quadro FX 3600M graphics card. The circle-example situation was used. We detail the total simulation loop time into the rendering and processing plots, which respectively correspond to the time spent during steps 1-3, and 4-6 (the latter includes the GPU-CPU data transfer).*

self-organized patterns of walkers emerge from the motion. Emergent patterns allow to efficiently solve the sum of interactions between walkers (cf. Table 1). Improvements compared to previous approaches are perceptible: in identical situations, the walkers *travel-time* is lowered using our approach, and the presence of *slow* walkers - with $v < 0.5m.s^{-1}$ and which may affect the overall believability of results - is decreased. In the example of the circle (Figure 7), other techniques concentrate walkers in the center of the scene which lower the efficiency of the circulation. In the case of the group-swap example, the Helbing's model fail to find an acceptable solution: groups are widely spread because particles simply repulse each-other.

	circle		group-swap	
	max. travel time	prop. of slow walkers	max. travel time	prop. of slow walkers
our model	53s.	0.97%	55s.	0.74%
Helbing's	90s.	30.4%	71s.	11.0%
RVO	63s.	13.0%	59s.	4.7%

Table 1: *The maximum walkers travel-time and the proportion of slow walkers are provided for the circle and the group-swap examples, using three different models. The proportion of slow walkers is the mean proportion of time walkers are going below* $0.5m.s^{-1}$.

Furthermore, a specificity of our model is to independently control angular and tangential velocity. Decelerations occur only in case of an imminent collision. The absence of deceleration during anticipated reaction results in smoother trajectories. We believe the overall aspect of our results is improved compared to previous approaches, especially when virtual humans are animated to follow the generated trajectories. The companion video illustrates the quality of synthetic trajectories, emergent self-organized patterns of walkers, as well as final animations.

5.2 Performances

Obtaining reasonable performance is probably the major technical challenge of the approach we propose due to the synthetic vision technique. We are still able to reach fair results by partly executing our algorithm steps on a GPU. Real-time performance (25 f.p.s.) is maintained up to 200 walkers (cf. Figure 11, computed on a laptop with Intel T7800@2.6GHz CPU and Quadro FX 3600M graphics card). The major bottleneck of our method is the data transfer from the GPU to the CPU (between steps 4 and 5).

Performance can be improved in several ways. Firstly, the camera resolution at step 1 can be lowered: on one hand, the number of perceived points is decreased accordingly and performance is improved; on the other hand, perception accuracy is decreased and may prevent walkers to react with anticipation to partly occluded obstacles. The companion video illustrates the impact of lowering the camera resolution. Secondly, we believe that the complete simulation loop can be executed on the GPU (recent approaches demonstrated feasibility [Silva et al. 2009]): on one hand, data-transfer between GPU and CPU is avoided (which represent approximately 30% of the complete simulation loop time); on the other hand, further tasks could be made impossible (e.g., animating virtual walkers). Finally, assuming that each obstacle is represented by a single static or moving points (which is, for instance, an acceptable assumption for a scene made of walkers only), the model applies without need to rely on synthetic vision. On one hand, interactions are directly considered between pairs of *moving points*, in place of each walker and a set of *perceived points*. The number of processed interactions is drastically lowered. But on the other hand, synthetic vision has many advantages: the visibility of obstacles walkers react to is implicitly checked, obstacles can have any 3D shape, walkers height - which may limit their perception - is taken into account, etc.

6 Discussion

Realism Our results demonstrate the ability of our approach to improve the emergence of self-organized patterns of walkers on several examples. From the standpoint of Computer Animation, our method provides visually appealing results. Interactions are solved more efficiently at the global-scale: compared to other approaches, the time required for walker to reach their goal is noticeably lower using our model (cf. Table 1 and companion video for comparisons). We believe the reached efficiency benefits to the resulting believability of animations, especially, some locking situations are avoided. It is however still required to quantitatively evaluate the realism of results. Studies based on spectators feedback or, better,

confrontation with real observations are possible directions for such an evaluation.

High-level behaviors and control Interactions between walkers are today limited to collision avoidance. Locomotion is controlled at the most basic level by visual-stimuli/motor-response laws. A near-future objective is to obtain a higher-level of control and to extend simulation abilities. Our first goal is to integrate some new types of interactions, such as following someone, reaching a mobile target, etc. Such interactions can easily be expressed in the $(\dot{\alpha}, tti)$-space. For instance, following p_i is controlling velocity so that $(\dot{\alpha}_i \rightarrow 0, tti_i \rightarrow cst)$ where cst is a positive constant. Then, our second goal is to combine different types of interactions to further improve the global efficiency of navigation by setting mid-term strategies (for instance, temporarily following someone is an efficient strategy to avoid further avoidance interactions) or to make possible the simulation of groups inside crowds (e.g., families). We assumed that goals were visible in our examples: a preliminary path planning stage would first be required to achieve navigation in complex environments. Path planners can decompose high-level goals into intermediary way-points that could be successively used as short-term goals in our model. A reactive change of short-term goals according to some external factors (e.g., local population densities) could be of interest: the evaluation future traffic conditions as well as the route selection process should be deduced from the visually perceived information in order to match our approach philosophy.

Model parameters Model's parameters (a, b, c) (cf. Equation (1)) can be adapted for each walker to individualize avoidance behavior with negligible computational overhead. The impact of parameters change on simulations is illustrated in the companion video. An intuitive link exists between avoidance behavior and the shape of τ_1 which is completely controlled by (a, b, c). The higher the peak of τ_1, the earlier the anticipation. The wider the peak, the stronger the adaptation. Finally, the curvature of τ_1 controls a trade-off between anticipation time and reaction strength: when the maximum curvature is higher, early anticipated reactions remain low whilst they get stronger when tti decreases. The automatic adaptation of parameters with respect to external factors, such as local density of population, may open interesting perspectives.

7 Conclusion

We presented a novel approach of crowd simulation made of individual walkers avoiding each other. Our main contribution is to steer walkers according to the visual perception they have of their environment. We formulate their collision avoidance behavior as visual-stimuli/motor-response control laws. Compared to previous vision-based approaches, we rely on statements from cognitive science that identified the visual-stimuli humans extract from their optic flow to control their locomotion and avoid obstacles. Compared to previous avoidance models, we demonstrate our approach improves the emergence of self-organized patterns of walkers in crowd simulations. In spite of the computational complexity raised by the synthetic vision technique, we demonstrate the ability of our approach to address complex interaction situations between numerous walkers. Our results are promising and open several future work directions. First is to automatically adapt the model parameters with respect to some external factors. A second direction is to extend our model to new types of interactions. Then, our objective is to add higher level of control in order to combine several types of interactions and to enable mid-term and long-term navigation strategies. Today, the proposed approach still results into visually interesting motions that can benefit to many Computer Animation applications. Evaluation of results by comparing them to real observations and data is now required. Nevertheless, our model is founded on cognitive science work on human locomotion which can open interesting perspectives for realistic simulation purposes.

References

CHAUMETTE, F., AND HUTCHINSON, S. 2006. Visual servo control, part i: Basic approaches and part ii: Advanced approaches. *IEEE Robotics and Automation Magazine 13(4)*, 82–90.

CHENNEY, S. 2004. Flow tiles. In *Proc. 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '04)*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 233–242.

CUTTING, J. E., VISHTON, P. M., AND BRAREN, P. A. 1995. How we avoid collisions with stationary and moving objects. *Psychological Review 102(4)* (October), 627–651.

GAYLE, R., MOSS, W., LIN, M. C., AND MANOCHA, D. 2009. Multi-robot coordination using generalized social potential fields. In *Proc. IEEE International Conference on Robotics and Automation (ICRA '09)*, 106–113.

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

HALPERIN, C., ANJYO, K., CIOROBA, M., KANYUK, P., REGELOUS, S., YOSHIDA, T., AND SALVATI, M. 2009. Crowd animation: Tools, techniques, and production examples. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 Courses*, ACM, New York, NY, USA, 1.

HEIGEAS, L., LUCIANI, A., THOLLOT, J., AND CASTAGNÉ, N. 2003. A physically-based particle model of emergent crowd behaviors. In *Graphicon 2003*.

HELBING, D., AND MOLNAR, P. 1995. Social force model for pedestrian dynamics. *Physical Review E 51*, 4282.

HUGHES, R. L. 2003. The flow of human crowds. *Annual Review of Fluid Mechanics 35*, 169–182.

KAPADIA, M., SINGH, S., HEWLETT, W., AND FALOUTSOS, P. 2009. Egocentric affordance fields in pedestrian steering. In *Proc. 2009 Symposium on Interactive 3D graphics and games (I3D '09)*, ACM, New York, NY, USA, 215–223.

KARAMOUZAS, I., HEIL, P., VAN BEEK, P., AND OVERMARS, M. H. 2009. A predictive collision avoidance model for pedestrian simulation. In *Motion in Games*, 41–52.

KUFFNER, J. J., J., AND LATOMBE, J. C. 1999. Fast synthetic vision, memory, and learning models for virtual humans. In *Proc. Computer Animation*, 118–127.

LAMARCHE, F., AND DONIKIAN, S. 2004. Crowds of virtual humans : a new approach for real time navigation in complex and structured environments. *Eurographics'04: Computer Graphics Forum 23*, 3 (September), 509–518.

LOSCOS, C., MARCHAL, D., AND MEYER, A. 2003. Intuitive crowd behaviour in dense urban environments using local laws. *Theory and Practice of Computer Graphics (TPCG'03)*.

MASSIVE. http://www.massivesoftware.com.

NARAIN, R., GOLAS, A., CURTIS, S., AND LIN, M. 2009. Aggregate dynamics for dense crowd simulation. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*.

NOSER, H., RENAULT, O., THALMANN, D., AND THALMANN, N. M. 1995. Navigation for digital actors based on synthetic vision, memory, and learning. *Computers & Graphics 19*, 1, 7–19. Computer Graphics Lab., Swiss Federal Inst. of Technol., Lausanne, Switzerland.

PARIS, S., DONIKIAN, S., AND BONVALET, N. 2006. Environmental abstraction and path planning techniques for realistic crowd simulation. *CASA 2006: Computer Animation and Virtual Worlds 17*, 3-4, 335.

PARIS, S., PETTRÉ, J., AND DONIKIAN, S. 2007. Pedestrian reactive navigation for crowd simulation: a predictive approach. *Eurographics'07: Computer Graphics Forum 26*, (3), 665–674.

PELECHANO, N., ALLBECK, J. M., AND BADLER, N. I. 2007. Controlling individual agents in high-density crowd simulation. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 99–108.

PELECHANO, N., ALLBECK, J., AND BADLER., N. I. 2008. *Virtual Crowds: Methods, Simulation, and Control.* Morgan & Claypool Publishers. (Series Editor: Brian Barsky).

PETERS, C., AND O'SULLIVAN, C. 2003. Bottom-up visual attention for virtual human animation. In *International Conference on Computer Animation and Social Agents (CASA'03)*, 111–117.

PETTRÉ, J., CIECHOMSKI, P. D. H., MAÏM, J., YERSIN, B., LAUMOND, J.-P., AND THALMANN, D. 2006. Real-time navigating crowds: scalable simulation and rendering. *Computer Animation and Virtual Worlds 17*, 3-4, 445–455.

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

REYNOLDS, C. W. 1987. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 25–34.

REYNOLDS, C. W. 1999. Steering behaviors for autonomous characters. In *Game Developers Conference 1999*.

SCHADSCHNEIDER, A. 2001. Cellular automaton approach to pedestrian dynamicstheory. In *In Pedestrian and Evacuation Dynamics*, 75–85.

SHAO, W., AND TERZOPOULOS, D. 2005. Autonomous pedestrians. In *Proc. 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '05)*, ACM Press, New York, NY, USA, 19–28.

SILVA, A. R. D., LAGES, W. S., AND CHAIMOWICZ, L. 2009. Boids that see: Using self-occlusion for simulating large groups on gpus. *Comput. Entertain. 7*, 4, 1–20.

SUD, A., ANDERSEN, E., CURTIS, S., LIN, M., AND MANOCHA, D. 2007. Real-time path planning for virtual agents in dynamic environments. *Proc. IEEE VR 2007*, 91–98.

THALMANN, D., AND RAUPP MUSSE, S. 2007. *Crowd Simulation*. Springer, London.

THALMANN, D., KERMEL, L., OPDYKE, W., AND REGELOUS, S. 2005. Crowd and group animation. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, ACM, New York, NY, USA, 1.

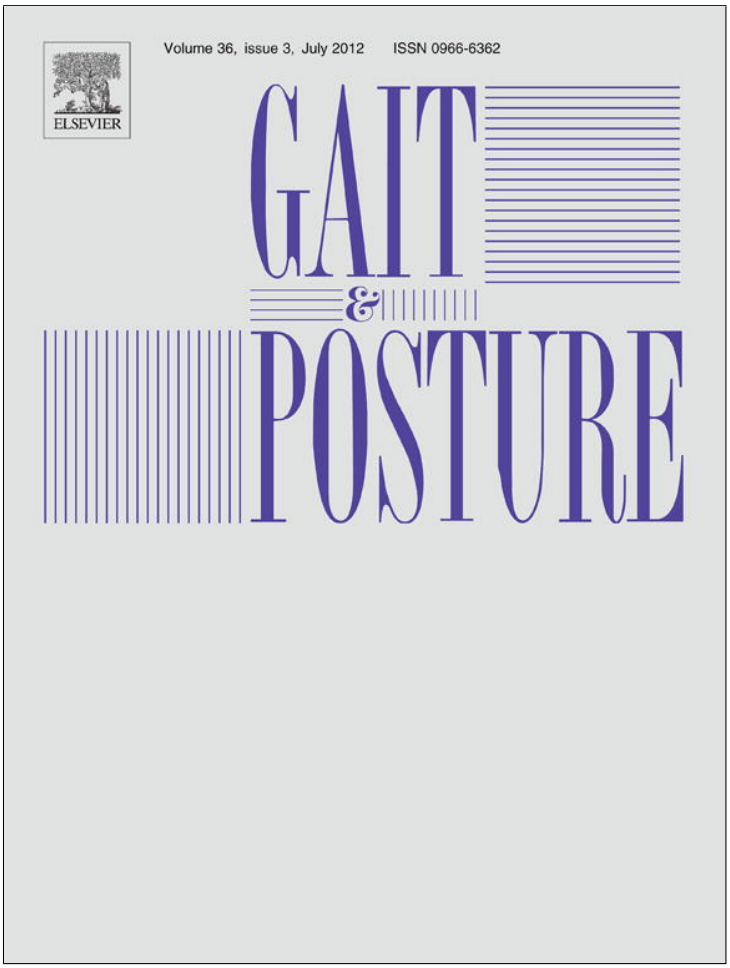
TRESILIAN, J. R. 1994. Perceptual and motor processes in interceptive timing. *Human Movement Science 13*, 335–373.

TREUILLE, A., COOPER, S., AND POPOVIĆ, Z. 2006. Continuum crowds. *ACM Transactions on Graphics (SIGGRAPH 2006) 25*, (3).

TU, X., AND TERZOPOULOS, D. 1994. Artificial fishes: physics, locomotion, perception, behavior. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 43–50.

VAN DEN BERG, J., PATIL, S., SEWALL, J., MANOCHA, D., AND LIN, M. 2008. Interactive navigation of individual agents in crowded environments. *Symposium on Interactive 3D Graphics and Games (I3D 2008)*.

WARREN, W. H., AND FAJEN, B. R. 2004. *Optic Flow and Beyond*. Kluwer (Editors: L. M. Vaina, S. A. Beardsley, and S. Rushton), ch. From optic flow to laws of control, 307–337.



Gait & Posture 36 (2012) 399–404

Contents lists available at SciVerse ScienceDirect

Gait & Posture

journal homepage: www.elsevier.com/locate/gaitpost

Minimal predicted distance: A common metric for collision avoidance during pairwise interactions between walkers

Anne-Hélène Olivier [a,b,*], Antoine Marin [a], Armel Crétual [a], Julien Pettré [b]

[a] Laboratoire Mouvement Sport Santé (M2S), UFRAPS, Université Rennes 2-ENS Cachan, Avenue Charles Tillon, CS24414, 35044 Rennes, France
[b] INRIA, Centre de Rennes Bretagne Atlantique, Campus Universitaire de Beaulieu, 263 avenue du Général Leclerc, 35042 Rennes, France

ARTICLE INFO

Article history:
Received 14 June 2011
Received in revised form 7 March 2012
Accepted 27 March 2012

Keywords:
Interaction
Locomotion
Collision avoidance
Gait adaptations
Anticipatory locomotor control

ABSTRACT

This study investigated collision avoidance between two walkers by focusing on the conditions that lead to avoidance manoeuvres in locomotor trajectories. Following the hypothesis of a reciprocal interaction, we suggested a mutual variable as a continuous function of the two walkers' states, denoted minimum predicted distance (*MPD*). This function predicts the risk of collision, and its evolution over time captures the motion adaptations performed by the walkers. By groups of two, 30 walkers were assigned locomotion tasks which lead to potential collisions. Results showed that walkers adapted their motions only when required, i.e., when *MPD* is too low (<1 m). We concluded that walkers are able (i) to accurately estimate their reciprocal distance at the time the crossing will occur, and (ii) to mutually adapt this distance. Furthermore, the study of *MPD* evolution showed three successive phases in the avoidance interaction: observation where $MPD(t)$ is constant, reaction where $MPD(t)$ increases to acceptable values by adapting locomotion and regulation where $MPD(t)$ reaches a plateau and slightly decreases. This final phase demonstrates that collision avoidance is actually performed with anticipation. Future work would consist in inspecting individual motion adaptations and relating them with the variations of *MPD*.

1. Introduction

When two humans walk in the same proximity, each can be considered as a moving obstacle for the other one. Such a situation occurs during everyday life activities such as walking in the streets. There is a fundamental difference between the avoidance of a non-human moving obstacle and the avoidance of human moving obstacles. The situation, in essence, is reciprocal: each walker is avoiding the other one while being avoided at the same time.

Previous studies have focused on the locomotion trajectory of a walker confronted with static or passive moving obstacles. Studies have mainly described the adaptation made to step over [1–3] or to circumvent [4] static obstacles. The extension to passive moving obstacles in a few studies has shown that walkers adapt their trajectories along both the anteroposterior and mediolateral axes to avoid a mannequin with a predefined trajectory [5–7]. The observed clearance area, also known as personal space, was modelled as an ellipse which dimensions depend on the level of attention required by the task [5]. In another study, Fajen and Warren [8] proposed to model interactions between a walker and the environment as a pair of coupled dynamic systems. Authors proposed to adapt heading according to the distance and the angle between the walker and stationary goals and obstacles. This model was extended to the avoidance of moving obstacles [9–11]. Following a vector–field interaction model in which goals represent attractors and obstacles represent repellors, the path of the walker was computed at each instant as the resultant of all forces applied to him/her. To the best of our knowledge, no study has considered the avoidance behavior between two human walkers. Two-human interactions have however been investigated by considering interpersonal coordination [12–14]. Ducourant et al. [12] focused on two participants (a leader and a follower) placed face to face and moving forwards and backwards. Results showed the presence of coordination mechanisms that depend on leadership and distance between people. This study provides an understanding of the interaction mechanisms during walking. However, trajectories were highly constrained and the nature of interactions between walkers was very specific. Compared to previous studies, our objective was to investigate collision avoidance between two human walkers. The main question was to identify the conditions that lead to avoidance manoeuvres in locomotor trajectories: what are the relations between the respective positions and velocities which yield motion adaptations? Based on the assumption of a reciprocal interaction, we

* Corresponding author at: INRIA, Centre de Rennes Bretagne Atlantique, Campus Universitaire de Beaulieu, 263 avenue du Général Leclerc, 35042 Rennes, France. Tel.: +33 2 99 84 71 38.
E-mail addresses: anne-helene.olivier@inria.fr (A.-H. Olivier), antoine.marin@univ-rennes2.fr (A. Marin), armel.cretual@univ-rennes2.fr (A. Crétual), julien.pettre@inria.fr (J. Pettré).

0966-6362/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.gaitpost.2012.03.021

suggested a mutual variable, common to both walkers, the minimum predicted distance (*MPD*), which (i) predicts potential collisions and (ii) describes the mechanism of collision avoidance over time in three successive phases.

2. Methods

2.1. Participants

Thirty participants (11 women and 19 men) volunteered for this experiment. They had no known vestibular or neurological pathology which would affect their locomotion. Participants gave written and informed consent before their inclusion and the study conformed to the Declaration of Helsinki. They were 26.1 years old (±6.9) (mean ± S.D.) and 1.74 m tall (±0.09).

2.2. Experimental protocol and apparatus

A 15 m edge square was used for the experiments. In pairs of two, starting from corners not sharing the same diagonal, participants were instructed to reach the opposite corner without restriction on gait or path. Interaction was mainly based on visual information: participants were not allowed to speak during the experiment and they walked barefoot or with socks on a carpet to avoid anticipation through auditory clues.

Participants waited for a start signal displayed on a computer screen placed on their right at each corner of the area. By synchronizing the two start signals, we provoked situations of potential collisions on orthogonal trajectories. The variability in natural speeds and reaction times actually changed the exact kinematic conditions of interactions, thereby allowing us to study their influence. The presence of occluding walls (2 m high by 3 m long) between corners (Fig. 1A and B) prevented participants from seeing each other before reaching their natural speeds. More precisely, there were six participants in a session located at the four corners of the area, but only two of them were actually given a start signal. This prevented walkers from anticipating who they would interact with and from which side he/she would come. 420 trials were performed.

2.3. Analysis

3D kinematic data were recorded using 12 Vicon MX-40 cameras (Oxford Metrics®) at a sampling rate of 120 Hz. Reconstruction and labeling were performed using Vicon IQ software and computations using Matlab (Mathworks®). We approximated participant's motions by using the middle point between their shoulders (two reflective markers were attached to participants acromions). The present study focused on the overall duration of the interaction between two walkers which lasted in average 4.1 s (±0.5). The higher frequency stepping oscillations were averaged out by applying a butterworth low-pass filter (dual-pass, third order, 0.5 Hz cut-off frequency) on mid-shoulder positions. Velocity was computed as the discrete time derivative of the mid-shoulder position in the horizontal plane.

2.3.1. Temporal segmentation

Experimental conditions prevented participants from seeing each other before they reached their natural speeds. By analyzing the geometry of occluding walls and the position of participants, we derived the time at which participants first saw each other (denoted '*tsee*'). They had orthogonal and convergent trajectories: they reached a minimum distance between them (clearance distance denoted '*dmin*') and we measured the time '*tcross*' at which *dmin* occurred (Fig. 1C). Crossing was considered as a relative concept in space (*dmin*) and time (*tcross*) between participants. We then focused on the analysis of the portion of data between *tsee* and *tcross*, given that interaction would occur during this period. We performed a temporal normalization of all trials between *tsee* (0%) and *tcross* (100%) to enable comparison.

2.3.2. Minimal predicted distance

We introduced and based our analysis on the minimal predicted distance (*MPD*): at each instant t, $MPD(t)$ represents the distance at which participants would meet if they did not perform motion adaptation after this instant t. Distance, being a mutual variable, appears relevant to describe reciprocal interactions. This distance was strictly positive since measured between the middle of the shoulders of each walker. When assuming that no motion adaption was performed, we can model future trajectories of walkers as linear extrapolations of their current states. For example, the trajectory of participant #1 was predicted by $P_{\text{pred},1}(t, u)$ as follows:

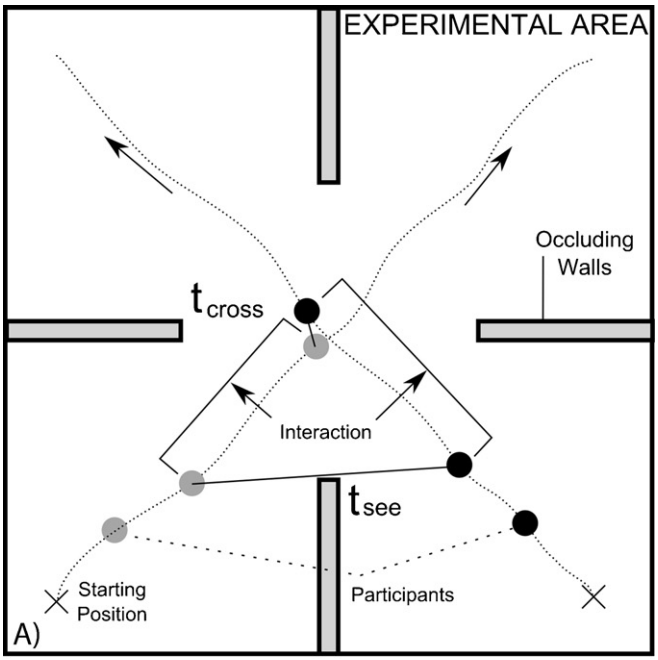
$$P_{\text{pred},1}(t,u) = P_1(t) + (u-t)V_1(t) \quad (1)$$

where u is a time parameter, $P_1(t)$ the current position and $V_1(t)$ the current velocity vector of participant #1.

MPD is thus formulated by computing the minimum distance between predicted positions $P_{\text{pred},1}$ and $P_{\text{pred},2}$ (Fig. 2A) reached by participants #1 and #2:

$$MPD(t) = \underset{u}{\operatorname{argmin}} \left\| P_{\text{pred},2}(t,u) - P_{\text{pred},1}(t,u) \right\| \quad (2)$$

Eq. (2) can be solved as the argument of the minimum of a second degree polynomial. If a positive solution is found ($u > 0$), trajectories are converging; if a



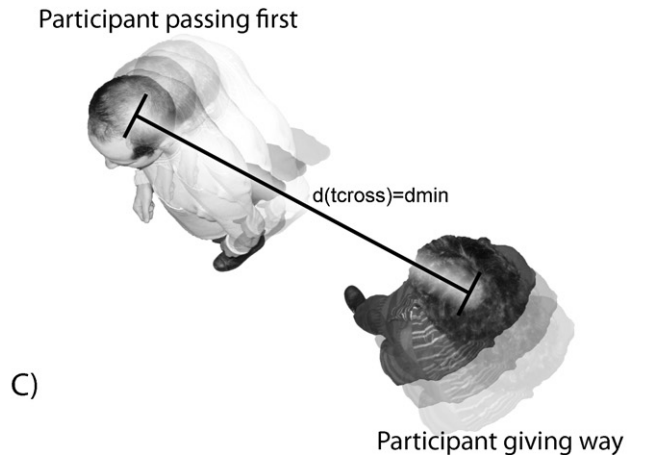


Fig. 1. (A) Experimental setup. Area is 15 m × 15 m. Two participants stand at the corners of the area and are synchronously given a start signal. Their task is to walk to the opposite corner. They implicitly start an interaction to avoid any collision. (B) Picture taken during experiment. (C) *tcross* is the time when the distance between walkers is minimal.

negative solution is found ($u < 0$), they are diverging; finally, if a null solution is found ($u = 0$), trajectories have actually reached their point of minimal distance. Indeed:

$$MPD(tcross) = d\min \quad (3)$$

dmin cannot be lower than a threshold distance considered to be admissible by participants and which needs to be precisely investigated. Nevertheless, we hypothesized its definition as a combination of contact distance (i.e., no collision between body envelopes) and social distance as suggested by previous studies on personal space preservation [5,15].

MPD is a prediction of *dmin* given current position and velocity of walkers. *MPD* varies if and only if motion adaptations are performed (Fig. 2B). We hypothesized that motion adaptations are linked to the admissibility of future clearance distance.

2.3.3. Statistics

Statistics were performed using Statistica (StatSoft®). The data were presented with mean and standard deviations. All effects were reported at $p < 0.05$. Wilcoxon signed-rank tests were used to determine differences between values of *MPD* at various instants.

3. Results

No collision occurred during the experiment and *dmin* was never below 0.41 m. Occluding walls fulfilled their role since participants reached a stable speed at *tsee*, i.e., before interaction. Fig. 3 illustrates a representative 90° crossing (A), the associated instantaneous velocity before *tsee* (B), and *MPD*(*t*) evolution during interaction (C). In this situation, the initial *MPD* (*MPD*(*tsee*)) is approximately 0.2 m (i.e., a future collision will occur if no adaptation is performed) and increases along the trial to reach 0.8 m at *tcross*.

Throughout all trials, the mean walking speed of participants was 1.57 m s^{-1} (±0.24) during the interaction phase and the mean clearance distance *dmin* was 1.09 m (±0.47) ranging from 0.41 to 3.48 m. *MPD*(*tsee*) ranged from 0 to 3.81 m (Fig. 4A). To consider the wide variety of *MPD*(*tsee*) values across our experiment, we subdivided the dataset in 10 groups of 42 trials according to

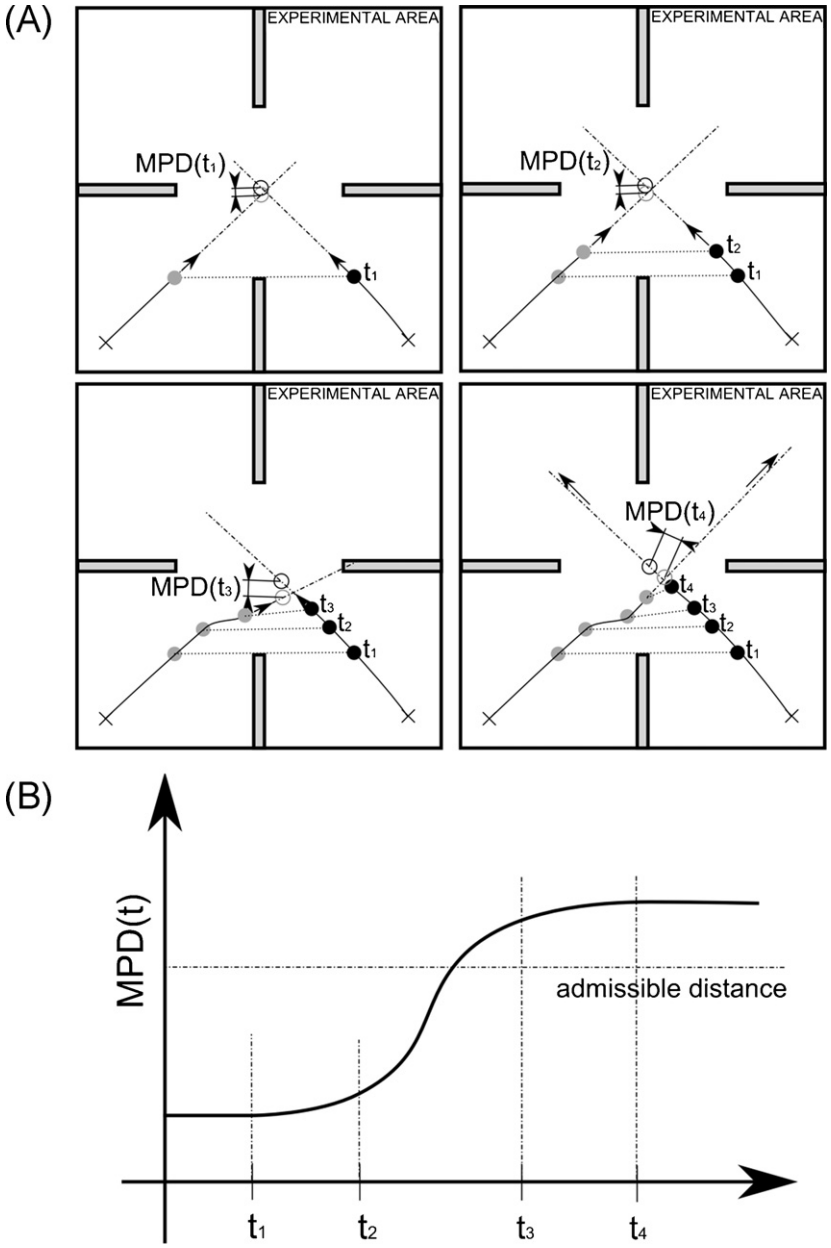


Fig. 2. (A) Schematic illustration of the minimum predicted distance (*MPD*) computed at four different times. A motion adaptation occurs between times *t*2 and *t*3. (B) *MPD*(*t*) evolution in time. *MPD* values change between *t*2 and *t*3: the increase of *MPD* indicates that motion adaptations were performed. In that case *MPD* was above an admissible distance at crossing.

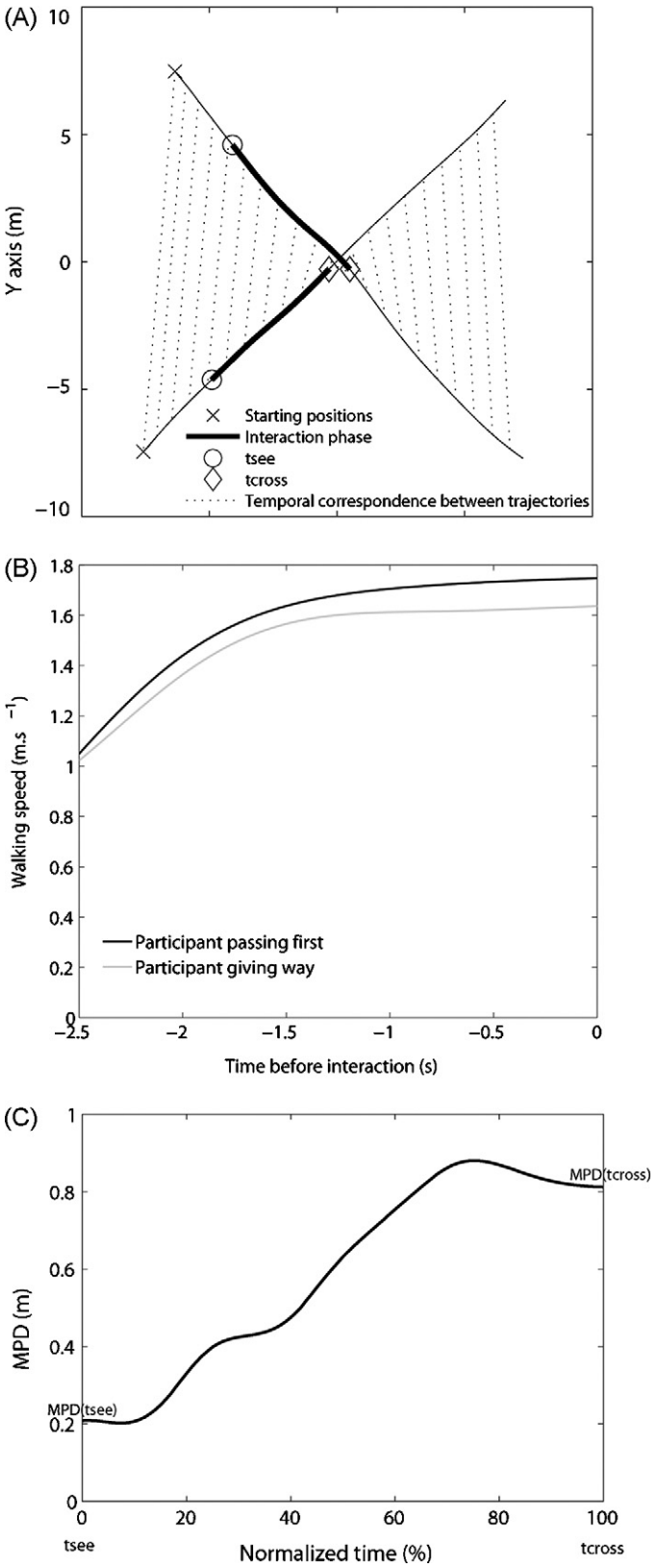


Fig. 3. (A) Participants' trajectories for one 90° crossing. Interaction phase is in bold line. (B) Instantaneous walking speed for both participants before *tsee*: they reach a stable speed before interaction. (C) *MPD*(*t*) evolution during the interaction phase.

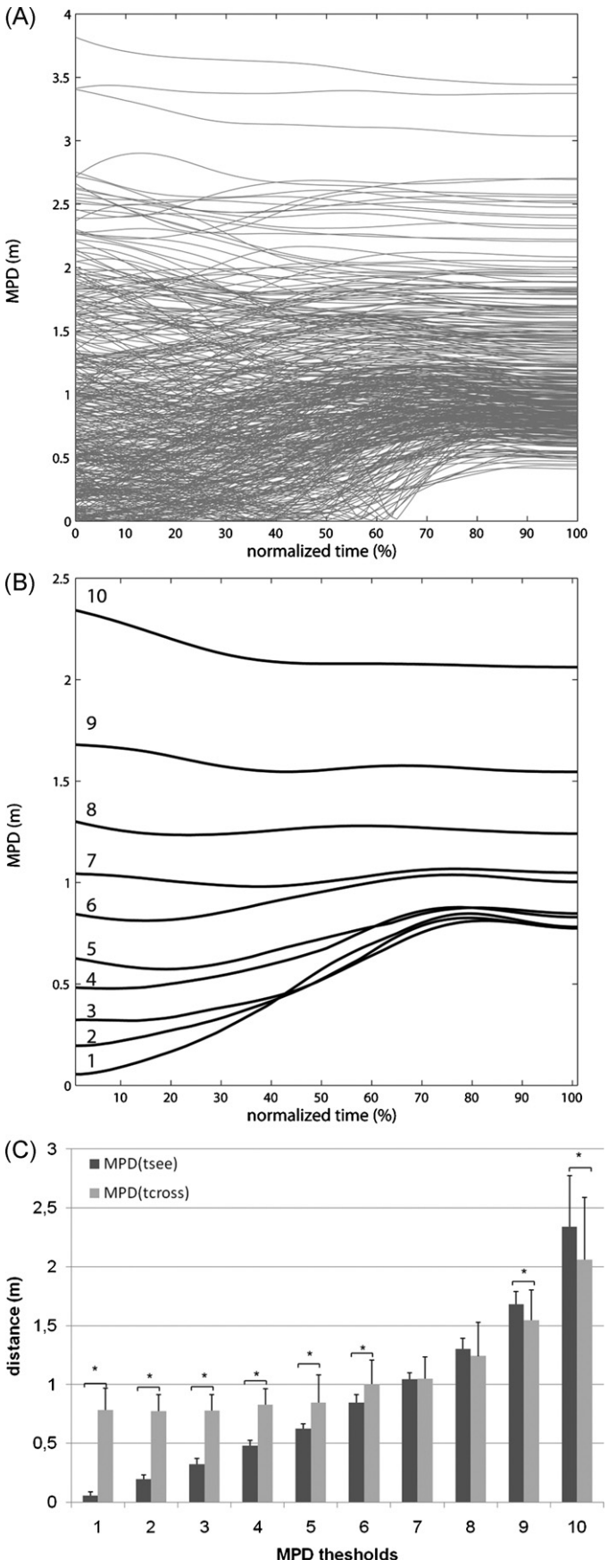


Fig. 4. (A) *MPD*(*t*) evolution for each of the 420 trials during the interaction phase. (B) Mean $\overline{MPD}(t)$ evolution for 10 groups of ascending *MPD*(*tsee*) values. (C) Mean values of *MPD*(*tsee*) and *MPD*(*tcross*) for each of the 10 groups.

ascending $MPD(tsee)$ values. For each group, we computed $\overline{MPD}(t)$, the mean MPD evolution along the interaction phase (Fig. 4B). When $\overline{MPD}(tsee)$ is lower than 1 m (groups 1–6), the set of $MPD(tcross)$ values for each group (see Fig. 4C) is significantly higher than $MPD(tsee)$ (respectively, Wilcoxon signed-rank tests results were $T_1 = 0$, $T_2 = 0$, $T_3 = 0$, $T_4 = 0$, $T_5 = 72$, $T_6 = 125$; $df = 41$, $p < 0.01$). When $\overline{MPD}(tsee)$ ranges from 1 to 1.5 m, there is no significant difference between the sets of $MPD(tcross)$ and $MPD(tsee)$ ($p > 0.05$). When $\overline{MPD}(tsee)$ is higher than 1.5 m, $MPD(tcross)$ is significantly smaller than $MPD(tsee)$ (respectively, $T_9 = 208$, $T_{10} = 56$; $df = 41$, $p < 0.05$).

Results showed that walkers adapted their trajectories to increase $MPD(t)$ when $MPD(tsee)$ was lower than 1 m. For all these trials, we computed the overall mean $\overline{MPD}(t)$ and its time derivative (Fig. 5A and B). We then considered three successive phases in time with respect to the value of the time derivative $\overline{MPD'}(t)$. The first phase, to which we referred to as the observation phase, was between normalized time $t0\%$ and $t7\%$, for which $\overline{MPD'}(t)$ was negative. Note that we still considered $\overline{MPD}(t)$ to be constant during the observation phase with respect to Wilcoxon signed-rank tests ($MPD(t0\%) = 0.44 \pm 0.28$, $MPD(t7\%) = 0.44 \pm 0.28$, $p > 0.05$). The second phase, from $t7\%$ to $t79\%$, was called the reaction phase: $\overline{MPD'}(t)$ was positive and $\overline{MPD}(t)$ significantly increased up to 0.88 ± 0.22 m ($T = 258$, $df = 263$, $p < 0.01$). Finally, the third phase, from $t79\%$ to $t100\%$, was called the regulation phase: $\overline{MPD'}(t)$ was negative again and $\overline{MPD}(t)$ slightly decreased to $dmin = 0.84 \pm 0.19$ m, ranging from 0.41 to 1.48 m ($T = -4648$, $df = 263$, $p < 0.01$). The mean trial duration was 4.1 s (± 0.5). Therefore, these three periods of time respectively lasted about 0.3 s, 3 s and 0.8 s.

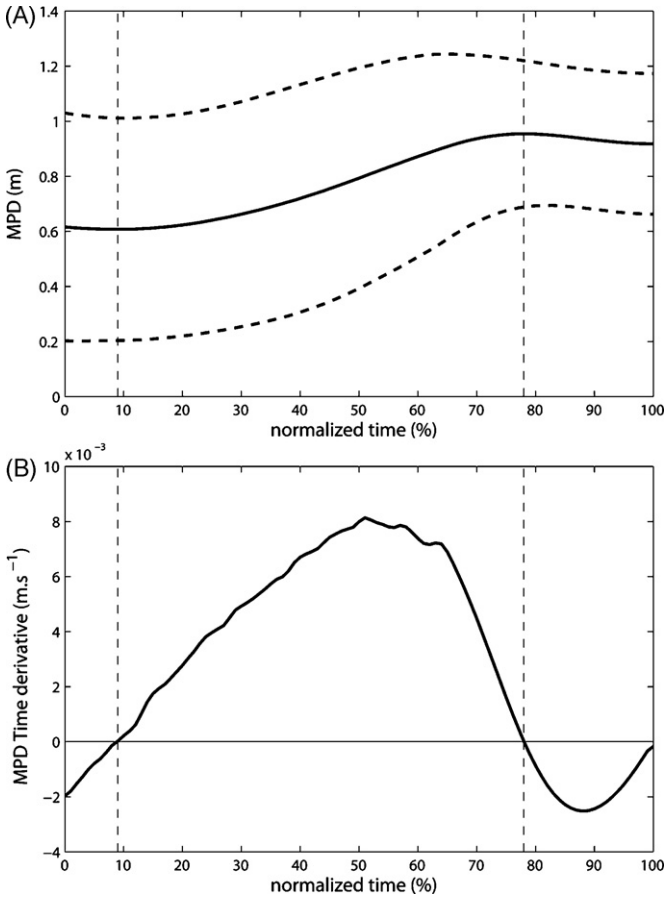


Fig. 5. Mean $\overline{MPD}(t)$ evolution ($\pm$S.D.) (A) and its time derivative (B) for all trials for which $MPD(tsee)$ is below 1 m. Interaction follows three successive phases: observation, reaction and regulation phases.

4. Discussion

We experimentally examined interactions between two participants avoiding each other. We then considered MPD as well as its variations as kinematic clues to represent interaction. Finally, we described collision avoidance as a three-phases process.

$MPD(t)$ is a predictive variable which is defined as the distance walkers would meet if no adaptation to their trajectories was performed. $MPD(t)$ varies in time if and only if locomotion is adapted by one or both walkers. An experimental setup allowed us to observe changing initial conditions of $MPD(tsee)$. By grouping trials according to $MPD(tsee)$ thresholds, typical behaviors were observed. When $MPD(tsee)$ is below 1 m, participants avoid a future collision by increasing this distance to reach an admissible value at $tcross$. When $MPD(tsee)$ is between 1 m and 1.5 m, no adaptation is performed and when $MPD(tsee)$ is above 1.5 m, participants even take the liberty of decreasing $MPD(t)$.

Based on the analysis of initial and final values of MPD, our general conclusion is that this variable is adapted only when required. Walkers are able to accurately estimate future crossing distance and to mutually adapt this distance. This result can be linked to the notion of personal space during collision avoidance between a walker and a mannequin mounted on an overhanging rail [5]. In our situation, the need to adapt trajectories is then revealed by $MPD(tsee)$. This mutual metric reveals the presence of motion adaptations, but does not relate individual collision avoidance strategies. Indeed, it was previously shown that walkers adapt their heading [8] or their heading and walking speed [5–7] to avoid a moving obstacle. Future work is then required to determine the nature of individual trajectory adaptations. Moreover, the locomotor path generated by the behavioral dynamic model [8] depended on the angle and the distance between the walker and the goals and obstacles. It would then be interesting to investigate the influence of these parameters on collision avoidance.

$MPD(t)$ also revealed the temporal structure of interactions. In the situation where the interaction requires motion adaptations ($MPD(tsee) < 1$ m), we identified three successive phases: observation, reaction, and regulation. Respectively, these phases correspond to periods of time when, first, $MPD(t)$ is constant, second, increases to acceptable values by motion adaptation and, third, reaches a plateau and slightly decreases. The observation phase is short. Information about future collision is quickly available. A similar observation was made by Gérin-Lajoie et al. [5] who observed an initial deviation of the trajectory one step after seeing a moving mannequin on a colliding path. The use of eye-trackers in future experiments would be a solution to study more carefully the characteristics of the observation phase as well as its duration in time. Gaze direction would additionally provide a more accurate description of the interaction by detailing visual information taken during the combined goal-directed and avoidance locomotion tasks.

The reaction phase is the longest part of the interaction. Participants adapt their trajectories to increase the future crossing distance and consequently to avoid a collision. There is no hesitation to the way interaction is mutually solved since $MPD(t)$ is increasing on average during this period of time. The value reached at the end of this phase is relatively constant (0.88 ± 0.21 m). These two observations show that participants accurately perceive the kinematics of the interaction and adapt their motion with positive effect on the interaction. Adaptation is quite optimal since $MPD(t)$ is not exaggeratedly increased at the end of this phase. This reaction phase can be linked to the anticipatory locomotor phase as described by Gérin-Lajoie et al. [5] during avoidance of dynamic obstacles.

Whereas this study analyzed individual adaptations, we focused on *MPD*(t) to illustrate that interactions are mutually solved. This variable does not indicate individual strategies but the effect of their joint combination.

The regulation phase, which starts approximately 0.8 s before crossing, demonstrates anticipation: the collision is solved, and the future crossing distance is maintained. It is even slightly decreased (4 cm). This anticipating behavior is not consistent with Fajen and Warren's model [8] but corroborates the results of other studies [5,16]. Moreover, contrasting with Gérin-Lajoie et al. [5], we did not observe a readjustment phase which increases the mediolateral distance from the obstacle before crossing. In their study, the mannequin was passive (i.e., with a linear trajectory), and therefore, the walker was not expecting any reaction. In our study, we considered interactions between two humans: they could expect a sharing of the effort to adapt trajectories, but with uncertainty about the other's attitude. This can explain that collision avoidance is solved 0.8 s in advance, close to the duration of a stride. This period may be associated with the one-stride interval related by Patla [17], which is sufficient to successfully implement adaptive strategies.

In conclusion, this study proposed a new metric, *MPD*(t), to investigate collision avoidance between two walkers. *MPD*(t) was defined as the prediction at each instant of the future crossing distance. Results showed that walkers adapt their motion only when required (when *MPD* is too low) with anticipation (existence of the regulation phase). Future work will investigate the nature and the quantity of individual adaptations necessary to solve interactions. The crossing order would be an important parameter since at the crossing point, the participant giving way views the other participant in front of him/her, and the participant passing first has the second one to his/her side or back. This asymmetric configuration emphasizes on asymmetric strategies for collision avoidance. Indeed, personal space may have an elliptic shape [5]. Therefore, collision risk should be perceived as being higher when the walker to avoid is in front compared to the side (see Fig. 1C). *MPD* is a relevant parameter to conduct such an analysis. First, *MPD* reveals the effect of individual reactions on the interaction, and second, the partial derivative of *MPD* with respect to each walkers speed and heading reveals the contribution of each participant manoeuvre to the evolution of *MPD*.

Acknowledgements

This research was funded by the ANR-PsiRob 2007 Locanthrope Project (ANR-07-ROBO-0007) and the European FP7-ICT-2009C Tango Project (n° 249858). We would like to thank Alain Berthoz and Marc Christie for helpful discussions.

Conflict of interest statement

None.

References

[1] Berard JR, Vallis LA. Characteristics of single and double obstacle avoidance strategies: a comparison between adults and children. Experimental Brain Research 2006;175(1):21–31.
[2] Krell J, Patla AE. The influence of multiple obstacles in the travel path on avoidance strategy. Gait and Posture 2002;16(1):15–9.
[3] Wang Y, Watanabe K. The relationship between obstacle height and center of pressure velocity during obstacle crossing. Gait and Posture 2008;27:172–5.
[4] Vallis LA, McFadyen BJ. Locomotor adjustments for circumvention of an obstacle in the travel path. Experimental Brain Research 2003;152:409–14.
[5] Gérin-Lajoie M, Richards C, McFadyen B. The negotiation of stationary and moving obstructions during walking: anticipatory locomotor adaptations and preservation of personal space. Motor Control 2005;9:242–69.
[6] Cinelli ME, Patla AE. Travel path conditions dictate the manner in which individuals avoid collisions. Gait and Posture 2007;26:186–93.
[7] Cinelli ME, Patla AE. Locomotor avoidance behaviours during a visually guided task involving an approaching object. Gait and Posture 2008;28:596–601.
[8] Fajen BR, Warren WH. A dynamical model of steering: obstacle avoidance, and route selection. Journal of Experimental Psychology-Human Perception and Performance 2003;29(2):343–62.
[9] Warren WH, Di S, Fajen BR. Behavioral dynamics of avoiding a moving obstacle. Journal of Vision 2003;3(9):134.
[10] Cohen JA, Bruggeman H, Warren WH. Switching behavior in moving obstacle avoidance. Journal of Vision 2005;5(8):312.
[11] Cohen JA, Bruggeman H, Warren WH. Combining moving targets and moving obstacles in a locomotion model. Journal of Vision 2006;6(6):135a.
[12] Ducourant T, Vieilledent S, Kerlirzin Y, Berthoz A. Timing and distance characteristics of interpersonal coordination during locomotion. Neuroscience Letters 2005;389(1):6–11.
[13] Oullier O, de Guzman GC, Jantzen KJ, Lagarde J, Scott Kelso JA. Social coordination dynamics: measuring human bonding. Society for Neuroscience 2008;3(2):178–92.
[14] Nessler JA, Gilliland SJ. Interpersonal synchronization during side by side treadmill walking is influenced by leg length differential and altered sensory feedback. Human Movement Science 2009;28(6):772–85.
[15] Hall ET. The hidden dimension. New York: Doubleday; 1966.
[16] Patla A, Tomescu S, Ishac M. What visual information is used for navigation around obstacles in a cluttered environment? Canadian Journal of Physiology and Pharmacology 2004;82(8–9):682–92.
[17] Patla AE. Understanding the roles of vision in the control of human locomotion. Gait and Posture 1997;5(1):54–69.

Avatar Locomotion in Crowd Simulation

N. Pelechano
U. Politècnica de Catalunya
abeacco@lsi.upc.edu

B. Spanlang
U. de Barcelona
bspanlang@ub.edu

A. Beacco
U. Politècnica de Catalunya
npelechano@lsi.upc.edu

Abstract

This paper presents an Animation Planning Mediator (APM) designed to synthesize animations efficiently for virtual characters in real time crowd simulation. From a set of animation clips, the APM selects the most appropriate and modifies the skeletal configuration of each character to satisfy desired constraints (e.g. eliminating foot-sliding or restricting upper body torsion), while still providing natural looking animations. We use a hardware accelerated character animation library to blend animations increasing the number of possible locomotion types. The APM allows the crowd simulation module to maintain control of path planning, collision avoidance and response. A key advantage of our approach is that the APM can be integrated with any crowd simulator working in continuous space. We show visual results achieved in real time for several hundreds of agents, as well as the quantitative accuracy.

Keywords: Crowd animation, Foot sliding.

1. Introduction

Crowd simulation in real time for computer graphics applications, such as training and video games, requires algorithms for not only agent navigation in large virtual environments while avoiding obstacles and other agents, but also rendering highly complex scenes with avatars to enhance realism. To achieve that, either of these steps can become a major bottleneck for real time simulation. Therefore, trade-offs between accuracy and speed are necessary. Simulating human motion accurately whilst keeping within constraints is not an easy task, and although many techniques have been developed for synthesizing the motion of one agent, they are not easily extended to large numbers of agents simulated in real time.

To achieve natural looking crowds, most of the current work in this area uses avatars with a limited number of animation clips. In most cases, problems arise when the crowd simulator module updates the position of each agent's root without taking into account movement of the feet. This yields unnatural simulations where the virtual humans appear to be 'skating' in the environment, which is known as 'foot sliding'. Other problems emerge from the lack of coherence between the orientation of the geometric representation of the agents and direction of movement. As we increase model sophistication through enhanced path selection, avoidance, rendering, and inclusion of more behaviors, these artifacts become more noticeable.

In order to avoid these problems, most approaches either perform inverse kinematics, which implies a high computational cost that does not allow large crowd simulations in real time, or adopt approaches where the animation clip being used drives the root position which limits movements and speeds.

In this paper we present an Animation Planning Mediator; a highly efficient animation synthesizer that can be seamlessly integrated with a crowd simulation system. The APM selects the parameters to feed a motion synthesizer while it feeds back to the crowd simulation module the required updates to guarantee consistency. Even when we only have a small set of animation clips, our technique allows a large and continuous variety of movement. It can be used with any crowd simulation software, since it is the crowd simulation module which drives the movement of the virtual agents and our module limits its work to adjusting the root displacement and skeletal state.

2. Related Work

We can classify crowd simulation models into two approaches. The first encompasses those models that calculate the movement of agents in a virtual environment without taking into consideration the underlying animation. The second is defined by those that have a core set of animation clips that they play back, or blend between, to move from one point to another.

The first group suffers from an artifact known as foot sliding, since the position of the characters is updated without considering the foot position. Notice this is not exactly the same problem addressed by other works removing motion capture noise [5]. In this group we have many examples using different crowd simulation models, such as social forces [4], rule-based models [13], cellular automata [17], flow tiles [1], roadmaps [16], continuum dynamics [18] and forces models parameterized by psychological and geometrical rules [12].

The second approach puts effort into avoiding foot-sliding while limiting the number of possible movements for each agent. Lau and Kuffner [8] introduced pre-computed search trees for planning interactive goal-driven animation. These models do not perform motion blending, are often limited to a small graph of animations and play one animation clip after another, thus moving the agent according to the root movement in each animation clip. As such they do not perform well in interactive, dynamic environments, especially with dense crowds where collision response forces have an impact.

Recent trends in character animation include driving physical simulations by motion capture data or using machine learning to parameterize motion for simplified interactive control. Examples are inverse kinematics and motion blending based on gaussian process statistics or geostatistics of motion capture data [2][11]. Such techniques avoid foot sliding but are computationally more expensive.

Through interpolation and concatenation of motion clips, new natural looking animations can be created [20]. Kovar et. al. introduced motion graphs [6]. Zhao et. al. extended them to improve connectivity and smooth transitions [23]. These techniques can avoid foot sliding by using the root velocity of the original motion data, but they require a large database of motion capture data to allow for interactive change of walking speed and orientation.

Ménardais et al. were able to synchronize and adapt different clips without motion graphs [10].

Proportional derivative controllers and optimization techniques are used [22] [14] to drive physically simulated characters. Goal-directed steps can perform a controlled navigation [21]. While such techniques show very impressive results for a single character in real time, the computational costs mean they are not suitable for real time large crowd simulations.

Kovar et. al. [7] presented an online algorithm to clean up foot sliding artifacts of motion capture data. The technique focuses on minute details and therefore is computationally not suitable for large real time crowds.

Treuille et. al. [19] generated character animations with near-optimal controllers using low-dimensional basis representation. This approach also uses graphs but, unlike previous models, blending can occur between any two clips of motion. They also avoid foot sliding by re-rooting the skeletons to the feet and specifying constraint frames, but their method requires hundreds of animation clips which is very time consuming to gather.

Recently, Gu and Deng [3] increased the variety realism creating new stylized motions from a small motion capture dataset. Maïm et. al. [9] apply linear blending to animations selected based on the agent's velocity and morphology achieving nice animations for crowds but without eliminating foot sliding.

3. The Framework

The framework employed for this work performs a feedback loop where the APM acts as the communication channel between a crowd simulation module and a character animation and rendering module. The outline of this framework is shown in Figure 1.

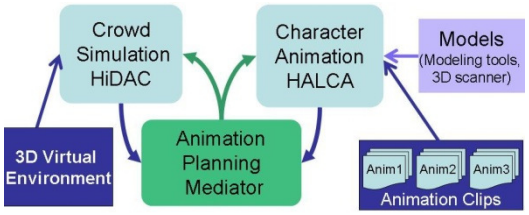


Figure 1 : Framework

For each frame, the crowd simulation module, CS, calculates the position p, velocity v, and desired orientation of the torso θ. This informa-

tion is then passed to the APM in order to select the parameters to be sent to the character animation module, CA, which will provide the next character's pose, $\mathcal{P}$. Each pose is a vector specifying all joint positions in a kinematic skeleton.

The APM calculates the next synthesized animation S_i which is described by the tuple $\{A_i, dt, b, \mathcal{P}, \mathbf{v}, \theta\}$ (see Table 1).

p	Agent root position.
$\mathbf{v}$	Agent velocity.
θ	Angle indicating torso orientation in x-z plane.
A_i	Animation clip for agent i.
dt	Differential of time for blending animation clip A_i
b	Blending factor when changing animation clip, i.e. when $A_i(t) \neq A_i(t-1)$, where t indicates time.
$\mathcal{P}$	Pose given by the skeletal state of the agent.

Table 1 : Input/Output variables of the APM

The APM may need to slightly adjust the agent's position and velocity direction in order to guarantee that the animations rendered are smooth and continuous. It is thus essential that the crowd simulation model employed works in continuous space and allows for updates of the position and velocity of each agent at any given time in order to guarantee consistency with the requirements dictated by the animation module. We have used the crowd simulation (CS) model HiDAC [12].

The torso orientation at time t, $\mathbf{w}_t$, is obtained from the velocity vector $\mathbf{v}_t$ after applying a filter so that it will not be unnaturally affected by abrupt changes.

$$\mathbf{w}_t = \omega_o \mathbf{w}_{t-1} + \mathbf{v}_t$$

where ω_o is the orientation weight introduced by the user and $\mathbf{w}_{t-1}$ is the direction of the orientation vector at time $t-1$. The orientation angle θ of the vector $\mathbf{w}$ is measured relative to the positive x-axis (since either the angle or the vector can be calculated from the other).

This orientation filter is applied as we want the position of the character to be able to react quickly to changes in the environment such as moving agents and obstacles, but we need the torso to exhibit only smooth changes, as without this the result will be unnatural animations where the rendered characters appear to twist constantly. Through filtering we can simulate an agent that moves with a slight zigzag effect, while the torso of the rendered character moves in a constant direction.

For the animation and visualization of avatars (CA) we are using a hardware accelerated library for character animation (HALCA)[15]. The core consists of a motion mixer and an avatar visualization engine. Direct access to properties such as the duration, frame rate, and the skeletal states of an animation are provided to the hosting application. Such information can be useful to compute, for example, the actual walking speed of a character when animated. Among the functionalities provided are: blending, morphing, and efficient access and manipulation of the whole skeletal state.

The CA contains a motion synthesizer which can provide a large variety of continuous motion from a small set of animations by allowing direct manipulation of the skeleton configuration. To create the library of animations, we decided to use hand created animations, although motion capture data could also be used after some pre-processing.

4. Animation Planning Mediator

To achieve realistic animation for large crowds from a small set of animation clips, we need to synthesize new motions.

The APM is used to find the best animation available while satisfying a set of constraints. On the one hand it needs to guarantee that the next pose of the animation will reflect as closely as possible the parameters given by the crowd simulation module (p, $\mathbf{v}$, θ), and on the other hand, it needs to guarantee smooth and continuous animations. Therefore, the selection of the best next character's pose needs to take into account the current skeletal state, the available animations, the maximum rotation physically possible for the upper body of a human, and whether there are any contact points to respect between the limbs of the skeleton and the environment (such as contact between a foot and the floor). Once the APM determines the best set of parameters for the next pose and passes this information to the CA for animation and rendering, it will also provide feedback to the CS in the cases where the parameters sent needed to be slightly adjusted to guarantee natural looking animations with the available set of animation clips and transitions.

During pre-processing the APM will calculate, for each animation clip average velocity $\mathbf{v}_{anim}$ in m/s by computing the total distance traveled by the character through the animation clip divided by the total duration of the animation clip, T, as well as the angle α between the torso orientation, θ_{anim}, and the velocity vector, $\mathbf{v}_{anim}$, in the animation clip. The i^{th} animation clip A_i is defined by the tuple $\{\mathbf{v}_{anim,i}, \alpha_i, T_i\}$.

During the simulation the APM takes the input parameters from the CS and proceeds through the five steps shown in Figure 2 to obtain the output tuple $\{Ai, dt, b, \mathcal{P}', p', \theta\}$ that will be sent to the CA. We explain this in detail below.

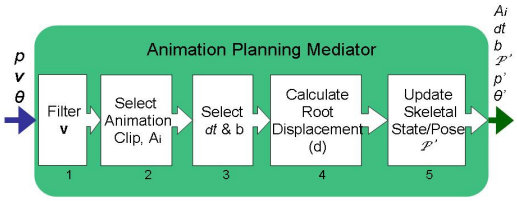


Figure 2 : Animator Planning Mediator

4.1. Animation Clip Selection

Instead of achieving different walking speeds by using hundreds of different walk animations, the algorithm can be effective with a limited number of animation clips by blending within an animation. This is given by the parameter *dt* which defines the time elapsed between two consecutive frames. Each animation clip covers a subset of speeds going from the minimum speed of 0 m/s (dt=0) to the original speed of the animation clip (dt=1).

An animation clip of walking forward can also be used to have the agent turn, as we can reorient the foot on the floor, thus reorienting the entire figure. This provides natural looking results for high walking velocities, but for slower velocities, using several turning animation clips and blending between them results in more natural looking motion.

If we consider α being the angle between the direction of movement v_{anim} and the torso orientation θ_{anim}, of an animation A_i, we can classify animations based on α and the velocity. For example, for an animation of walking sideways $\alpha = 90$ degrees and for walking forwards $\alpha = 0$ degrees.

To determine when to use each animation we classify them in a circle defined by tracks and sectors. A track is the area contained between two concentric circles. Each concentric circle has as radius the velocity of an animation, $\mathbf{v}_{anim}$. Once the tracks are defined we divide them into sectors, each of which maps to a clip.

All the animations used must satisfy the following requirements: (1) must be time aligned, (2) $\mathbf{v}$ and α must be approximately the same throughout the animation clip (within a small threshold defined by the user), and (3) animation clips must be cyclical.

Each animation clip could be used when the velocity of the agent $\mathbf{v} \leq \mathbf{v}_{anim}$, therefore we decide on which animation to assign to each sector depending on the α value. The decision points of when to switch from one animation sector to the next as the angle increases is defined as being halfway between the α values of two neighboring animations. Figure 3 graphically represents the decision framework, where colors are used to represent the animation clips assigned per sector. We have chosen some animation clips with similar velocities or angles ($v_3 \approx v_4$, $v_6 \approx v_7 \approx v_8$, and $\alpha_1 \approx \alpha_2 \approx \alpha_5$) to graphically show the splitting of tracks into sectors. Only half of the circle of animation clips is shown due to symmetry. At any time during the simulation we can run any animation backwards by using a negative dt and selecting the clip by flipping vertically over the x axis and mirroring in the y axis.

4.1.1 Classifying Motion Clips

Initially the algorithm starts by dividing the circle into tracks $T_1, T_2, \ldots, T_m$, where each track is defined by the velocity of an animation clip from the library (v_{anim}) and $v_1 > v_2 > \ldots > v_n$. Note that $m \leq n$ as two animations A_i and A_j with similar velocity (i.e. $|v_i - v_j| < \varepsilon_v$) will be assigned to the same track. Each track T_i is defined by its maximum and minimum velocity, $T_i = \{v_i^{min}, v_i^{max}\}$.

Starting from the outer track T_1 (highest velocity), the algorithm proceeds by splitting each track into sectors based on the γ values defined as being halfway between the α values of two animations.

Each sector $S_{i,k}$ inside track T_i is defined by the tuple: $\{v_i^{min}, v_i^{max}, \gamma_{i,k}^{min}, \gamma_{i,k}^{max}\}$ which corresponds to the minimum and maximum velocity, and minimum and maximum decision angles.

For each step of the algorithm, a track T_{i+1} will be split into at least as many sectors as contained within T_i. For each sector $S_{i,k} = \{v_i^{min}, v_i^{max}, \gamma_{i,k}^{min}, \gamma_{i,k}^{max}\}$ there will be a new sector $S_{i+1,k} = \{v_{i+1}^{min}, v_{i+1}^{max}, \gamma_{i,k}^{min}, \gamma_{i,k}^{max}\}$ where $v_{i+1}^{max} = v_i^{min}$, with

the same animation being assigned. Then further splitting of those sectors will occur for each of the new animations with $|v_{anim} - v_{i+1}^{max}| < \varepsilon_v$ depending on the α values of the remaining animations. If we let α_p be the angle of a new animation A_p and α_q be the angle of the previous animation A_q that is assigned to a new sector $S_{i+1,k}$, then if $|\alpha_p - \alpha_q| < \varepsilon_\alpha$, the new animation A_p replaces A_q for that sector. For any other case a new sector is created and the angle limits γ between sectors are recalculated.

The variety of movements that can be synthesized with this method will depend upon the number of animation clips. The more clips we have, the more sectors we can define.

During run time the APM will select the best animation clip based on the current velocity, **v**, of the agent, and the angle, φ, between the velocity and the torso orientation, θ, provided by the CS. If a change of animation is required, then the APM will determine the weight, b, necessary for blending between animations.

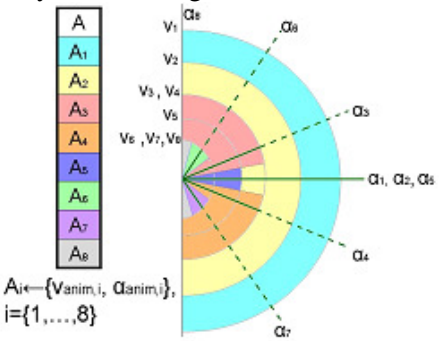


Figure 3 : Circular Decision Framework.

4.2. Blending Factors

At every frame the CS calculates the new position for each agent based on the desired path to move between an initial position to a destination, while interacting with other agents, walls and obstacles. However, since the CS is not aware of the type of animation being used for the rendering, if we take that new position to translate the root of the skinned avatar and the angle θ to reorient it, we will observe that the figures appear to 'skate', and also that there is no coherence between the orientation of the avatar and the actual animation movement.

To avoid foot-sliding and guarantee that the animation satisfies the constraints given by **v** and θ, we need to ensure that the figure appears to move according to **v** while the foot currently in contact with the floor stays in place, and the torso faces the direction given by θ. This could be done using inverse kinematics, but since we are simulating large crowds, we need a method that can be quickly calculated and applied to hundreds of 3D animated figures in real-time. Also, to avoid the time and cost of generating a large number of animation clips, we are interested in a limited number of clips that can be combined to achieve as many realistic animations as possible. We have a trade-off between the accuracy of our animations and the simplicity, and therefore speed, of our calculations.

Knowing the agent's velocity **v** and the animation velocity $\mathbf{v}_{anim}$ from the selected animation A_i, we can calculate the blending factor τ.

$$\tau = \frac{\mathbf{v}}{\mathbf{v}_{anim}} \quad , \qquad \tau \in [0,1]$$

The dt needed by the CA module to blend between poses is:

$$dt = \tau \cdot \Delta t$$

where Δt is the elapsed time between consecutive frames. At this point of our algorithm we have the new pose of the agent and thus can obtain the local coordinates of the root and the feet position.

Knowing that the root movement is driven by the foot that is on the floor during two consecutive poses, we update the root position in the global coordinates of the environment.

4.3. Calculation of Root Displacement

For the current pose $\mathcal{P}_t$, we calculate the vector $\mathbf{u}_t$ that goes from the foot on the floor to the root of the skeleton. Likewise for the pose $\mathcal{P}_{t-1}$ in the previous frame we calculate $\mathbf{u}_{t-1}$ (see figure 4). The vector $\mathbf{u}_t$ contains all the rotations that happen at the ankle and knee level and thus provides sufficient information about the leg movement.

By subtraction of vectors we can calculate root displacement **d** between frames. The vector of displacement **d**, shown in red is:

$$\mathbf{d} = \mathbf{u}_t - \mathbf{u}_{t-1}$$

And the new position is thus:

$$p_t = p_{t-1} + \mathbf{d}$$

The method is efficient enough to allow for fast calculation and extension to 3D is straight forward. From the results shown in Figure 5 we can observe that it avoids foot sliding and preserves the vertical root movement that appears when we walk fast. In the figure we have rendered the root path in black.

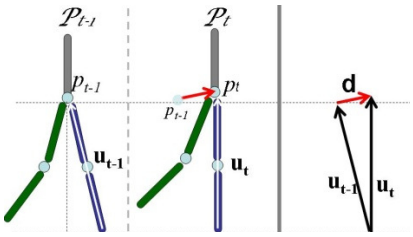


Figure 4 : Root displacement.

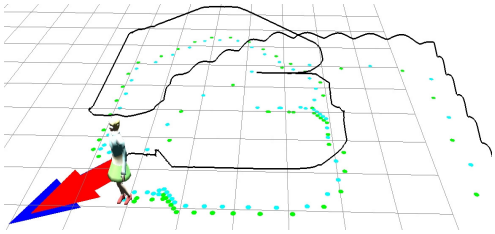


Figure 5 : Path followed by one agent.

4.4. Updating the Skeletal State

Turns in the environment can be achieved by reorienting the figure according to the velocity vector **v** and adjusting the torso orientation according to θ by modifying the skeletal configuration. This will also orient the displacement vector to move the character in the direction indicated by the CS module.

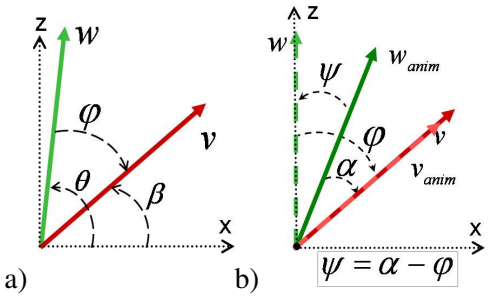


Figure 6 : Angles described in Table.

The visual effect of this is that the foot on the floor will slightly rotate in place. This is almost unnoticeable to the human eye, especially at high velocities, and thus is a trade-off worth considering as we can achieve turns in any direction without the requirement of having a large database of animation clips.
For slower velocities, we can achieve higher realism by having a number of simulations where the torso orientation does not necessarily need to be aligned with **v**. To calculate the rotations we use the following variables:

w	torso orientation vector.
β	angle in the x-z plane of the velocity vector **v** relative to the positive x-axis.
φ	angle of the velocity vector, **v,** relative to the orientation vector, **w**.
α	angle in the x-z plane of the velocity vector $\mathbf{v}_{anim}$ of the animation clip A_i relative to its orientation vector $\mathbf{w}_{anim}$.

Table 2 : Variables required for upper body rotation.

To achieve movement in the direction given by **v**, the avatar is rotated by $(\beta - \alpha)$ so that the velocity vector of the animation $\mathbf{v}_{anim}$ matches **v**. Then the torso is oriented according to **w**: the angle ψ is calculated as the difference between φ and α (Figure 6) and propagated across the spine of the character by modifying the current skeletal state of the character's pose. This allows the agent to move the root in the direction indicated by **v** while the torso is facing the direction indicated by **w**.

4.5. The APM Algorithm

The following table summarizes the APM algorithm with references to the sections where each step was explained:

Algorithm: APM	Section
$\varphi \leftarrow AngleBetween(\mathbf{w},\mathbf{v})$ $A_i \leftarrow SelectAnimation(\mathbf{v},\varphi)$	*4.1*
if $(A_i \neq A_{i-1})$ *then* $b \leftarrow Agent.BlendingFactor()$ $dt \leftarrow CalculateDT(\mathbf{v}, A_i)$	*4.2*
$\mathcal{P}_t \leftarrow getNextPose(A_i, \mathcal{P}_{t-1}, dt)$ $\mathbf{d} \leftarrow CalculateDisplacement(\mathcal{P}_t, \mathcal{P}_{t-1})$	*4.3*
$\alpha \leftarrow GetAngleAnim(Ai)$ $\beta \leftarrow CalculateAngle(\mathbf{v})$ $\psi \leftarrow \alpha - \varphi$ $agentCA.PropagateAngleSpine(\mathcal{P}_t, \psi)$	*4.4*

Algorithm 1 : The APM Algorithm

With the parameters calculated by the APM, we apply an absolute rotation of $(\beta - \alpha)$ and add the displacement, d, to the previous position to render our character satisfying the requirement given by the CS.

5. Results

The method presented only needs a small set of animation clips to obtain visually plausible results. By increasing the number of animations and/or the quality of those animations (i.e. using

motion capture data), we would obtain improved results with no additional cost during simulation time.
The animation library shown in the accompanying videos consists of four walking forward animations, two side-step animations and four "walking on an angle" animations.
Per agent and per frame, on a Intel Dual Core 3GHz with 4GB of RAM, the root displacement computation is less than 0.8µs. The whole APM algorithm requires less than 0.021ms. Therefore, we could incorporate the APM into any crowd simulation module, with an additional linear cost per frame of 0.021 ms times the crowd size. Since typically not all the agents in a crowd are visible simultaneously, we could apply the APM algorithm to only the few hundred agents closest to the camera.

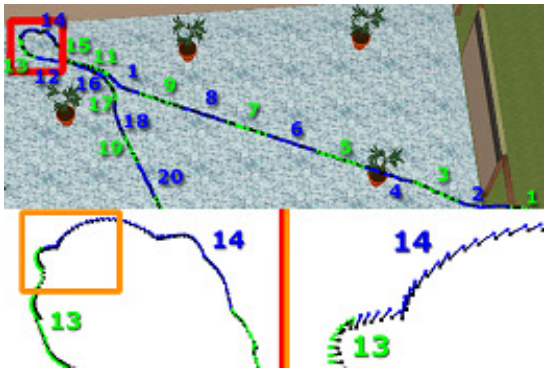


Figure 7 : Path followed by an agent with close-ups of a turn.

In order to satisfy the constraints given by the CA, our APM may need to slightly modify the position of the root given by the CS. Figure 7 shows the path followed by an agent, with a zoomed view of a sharp. Each blue/green segment corresponds to 75 frames of the animation (3 seconds). We have represented the deviation introduced by the algorithm as a segment with a green/blue ending indicating the position given by the CS and a black one indicating the corrected position calculated by the APM.
Deviation is bigger where there are abrupt turns and blending simultaneously. We have calculated the average deviation between the CS position and the APM corrected position in order to determine the impact of our algorithm on the final path. Running at 25fr/s, on average we obtain a deviation of less than 7.78mm per frame. Larger deviations correspond to segments 13 and 14 which are when the agent is turning sharply (Figure 8).

In Figure 9 we can see that deviations are larger at turns and reach maximum values when there are repulsion forces between agents (for example at the doors where agents congregate). Most of the time the deviation stays under 1cm per frame.

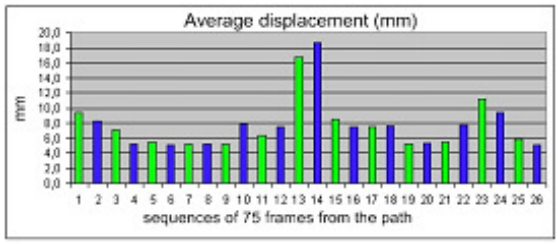


Figure 8 : Deviations in mm for each segment of the path shown in figure 7.

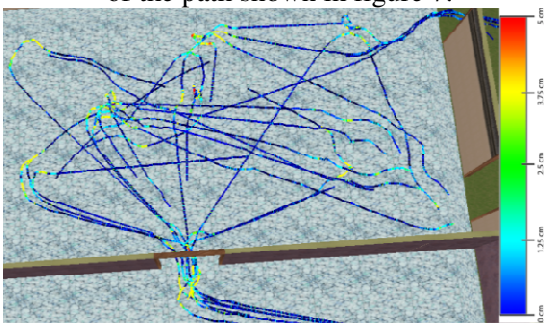


Figure 9 : Paths for 20 agents showing the deviation with a color gradient scale.

6. Conclusions

We present a realistic, yet computationally inexpensive, method to achieve natural animation without foot-sliding for crowds. Our goal was to free the crowd simulation module from the computational work of achieving natural looking animations and focus instead on developing crowd behavior that looks realistic. Natural looking results can be obtained with a minimal library of animation clips and this method can be integrated with any crowd simulation software.
As our model requires a foot on the floor to calculate the root displacement, it has some limitations. Currently we cannot achieve running simulations where both feet are in the air for certain frames. We plan on addressing this problem in future work.
The library employed for this work was hand created, and thus our original animations suffer from rigidity which affects the overall look of the crowd. Since the quality of the final crowd animation depends strongly on the set of animation clips available, having time aligned motion

capture animation clips will achieve more natural looking results.

7. Acknowledgements

This research has been funded by the Spanish Government grant TIN2010-20590-C0-01 and the ERC advanced grant TRAVERSE 227985.
A. Beacco is also supported by the grant FPU-AP2009-2195 (Spanish Ministry of Education).

References

[1] S. Chenney, "Flow Tiles". In. Proc. ACM Symposium on Computer Animation, 233–242. 2004

[2] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. "Style-based inverse kinematics". In ACM Transactions on Graphics, 23(3), 522–531. 2004

[3] Q. Gu and Z. Deng, "Context-Aware Motion Diversification for Crowd Simulation," In IEEE Computer Graphics and Applications. 2010

[4] D. Helbing, I. Farkas, and T. Vicsek. "Simulating Dynamical Features of Escape Panic", In Nature, 407, 487-490. 2000

[5] L.Ikemoto, O.Arikan, and D. Forsyth, "Knowing when to put your foot down". In ACM Proceedings of the 2006 symposium on Interactive 3D graphics and games (I3D '06). 49-53. 2006

[6] L. Kovar, M. Gleicher, and F. Pighin. "Motion Graphs". ACM Transactions on Graphics 21, 3, 473-482. 2002

[7] L. Kovar, J. Schreiner, and M. Gleicher, "Foot skate cleanup for motion capture editing" ACM Proc. Symp. on Computer Animation. ACM Press. 97-104. 2002

[8] M. Lau and J. Kuffner, "Precomputed Search Tree: Planning for Interactive Goal-Driven Animation". Proc. Symp. on Computer Animation. 299-308. 2006

[9] J. Maïm, B. Yersin, J. Pettré, and D. Thalmann, "YaQ: An Architecture for Real-Time Navigation and Rendering of Varied Crowds". IEEE Computer Graphics and Applications, 29(4), 44-53. 2009

[10] S. Ménardais, R. Kulpa, F. Multon and B. Arnaldi. "Synchronization for dynamic blending of motions". In. Proc. ACM Symposium on Computer Animation, 325–336. 2004

[11] T. Mukai and S. Kuriyama. "Geostatistical motion interpolation". In ACM Transactions on Graphics, 24(3), 1062–1070.2005

[12] N. Pelechano, J.M. Allbeck, and N.I. Badler, "Controlling Individual Agents in High-Density Crowd Simulation", ACM Proc. Symp. on Computer Animation.99-108. 2007.

[13] C. Reynolds. "Flocks, Herds, and Schools: A Distributed Behavior Model", In Proc. of ACM SIGGRAPH, 25-34.1987.

[14] M. da Silva, Y. Abe, and J. Popović. "Simulation of human motion data using short-horizon model-predictive control". In. Computer. Graphics. Forum, 27(2), 371–380. 2008.

[15] B. Spanlang. Halca hardware accelerated library for character animation. Technical report. Event Lab, Universitat de Barcelona. event-lab.org. 2009.

[16] A. Sud, E. Andersen, S. Curtis, M. Lin, and D. Manocha,"Real-time Path Planning for Virtual Agents in Dynamic Environments" In. Proc. IEEE Virtual Reality, 91–98. 2007.

[17] F Tecchia, C. Loscos, R. Conroy, and Y. Chrysanthou. "Agent behavior simulator (ABS): A Platform for Urban Behavior Development", In Proc. of ACM Games Technology Conference, 17–21. 2001.

[18] A. Treuille, S. Cooper, and Z. Popović. "Continuum Crowds" In. Proc. ACM Transactions on Graphics SIGGRAPH, 1160–1168. 2006.

[19] A. Treuille, Y. Lee, and Z. Popović, "Near-optimal Character Animation with Continuous Control", Proc. ACM Siggraph, 26(3), Article 7. 2007.

[20] A. Witkin, and Z. Popović, "Motion Warping", In. Proc. of Siggraph'95. 105-108. 1995.

[21] C.C., Wu, and V. Zordan, "Goal-directed stepping with momentum control", ACM Symp. on Computer Animation. 2010.

[22] K.K.Yin, K. Loken, and M. van de Panne. "Simbicon: Simple Biped Locomotion Control". In ÁCM Transactions on Graphics, 26(3), Article 105, 2007.

[23] L. Zhao and A. Safanova, "Achieving Good Connectivity in Motion Graphs", ACM Proc. Symp. on Computer Animation.139-152. 2008.

CEIG - Spanish Computer Graphics Conference (2013)
M. Carmen Juan and Diego Borro (Editors)

Dynamic Footsteps Planning for Multiple Characters

A. Beacco[1], N. Pelechano[1] & M. Kapadia[2]

[1]Universitat Politècnica de Catalunya
[2]University of Pennsylvania

Abstract

Animating multiple interacting characters in real-time dynamic scenarios is a challenging task that requires not only positioning the root of the character, but also placing the feet in the right spatio-temporal state. Prior work either controls agents as cylinders by ignoring feet constraints, thus introducing visual artifacts, or use a small set of animations which limits the granularity of agent control. In this work we present a planner that given any set of animation clips outputs a sequence of footsteps to follow from an initial position to a goal such that it guarantees obstacle avoidance and correct spatio-temporal foot placement. We use a best-first search technique that dynamically repairs the output footstep trajectory based on changes in the environment. We show results of how the planner works in different dynamic scenarios with trade-offs between accuracy of the resulting paths and computational speed, which can be used to adjust the search parameters accordingly.

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

1. Introduction

Animating groups of human characters in real time is a difficult but necessary task in many computer graphics applications, such as video games, training and immersive virtual environments. There is a large amount of work in the crowd simulation and pedestrian dynamics literature, but most applications still lack convincing character animation that offer a variety of animation styles without noticeable artifacts.

Humans walking in the real world have a cognitive map of the environment which they use for calculating their path through waypoints (doors, corners, etc), Then, we navigate along the path by choosing footsteps to avoid collisions with nearby humans and obstacles. Likewise, a virtual character can be simulated within an environment by first deciding a high level path (sequence of waypoints) using a navigation mesh [Mon09] [OP13] and then calculating the exact trajectory to walk from one waypoint to the next one. That trajectory is going to be defined by the chosen steering behavior algorithm, the output of which is going to encode the state of the agent over time. An agent state can be modeled by different granularities going from a simple point and radius with a velocity vector in a low level representation, to a complete high resolution mesh with joint velocity vectors, rotational angles, torques and any other elements that might improve the simulation on a higher level representation. Intermediate representations [SKRF11] can perform simulations in real-time by using an inverted pendulum model of the lower body of a biped which can be controlled to generate biomechanically plausible footstep trajectories.

This paper focuses on the computation of natural footsteps trajectories for groups of agents. Most work in the literature uses crowd simulation approaches (rules based models, social forces, cellular automata models, continuum forces) to calculate the root displacement between two consecutive waypoints. This leads to smooth root trajectories, but with many artifacts due to lack of constraints between the feet and the floor. There are some approaches that do focus on correct foot placement, but in most cases they are quite limited in the range of animations available or else can only deal with a small number of agents. Our work enforces foot placement constraints and uses motion capture data to produce natural animations, while still meeting real-time constraints for many interacting characters.

Figure 1 illustrates an example of four agents planning their footstep trajectory towards their goal while avoiding collision with other agents, and re-planning when necessary. The resulting trajectories not only respect ground contact

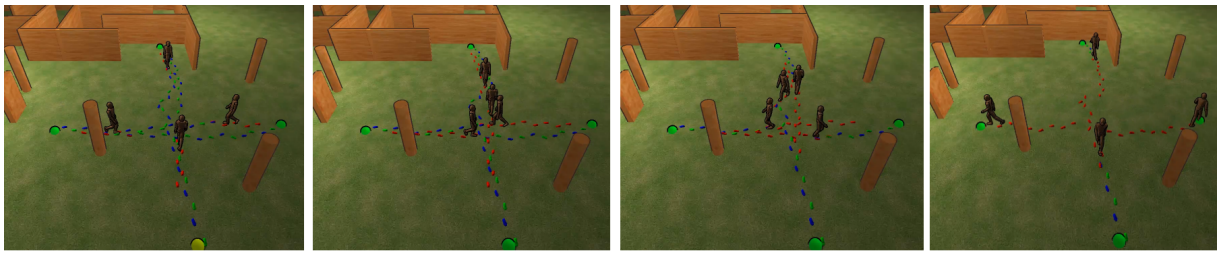


Figure 1: *Footstep trajectories planning for four agents reaching goals in opposite directions*

constraints, but also create more natural paths than traditional multi agent simulation methods.

This paper is organized as follows. We first examine previous approaches in crowd simulations and their methods. Next we give an overview of our framework and explain in detail our pre-process step, planning algorithm and animation system. Finally we show some of our results and present a discussion about the strength of our method and its limitations along with conclusions and future work.

2. Related Work

Crowd simulation approaches can be classified into two main sets based on whether they only focus on calculating the position of the root ignoring the animations, or whether they plan respecting the underlying animations. The first set focuses on simulating realistic behaviors regarding overall character navigation and do not worry about animations. In fact sometimes their goal is to simply model agents as cylinders that move around a virtual environment avoiding collisions. The second set, which carries out planning while being aware of the animation clips available, need to perform some pre-process to analyze the set of animation clips available to plan paths respecting constraints between the feet and the floor. In some cases, if the animation set is handmade, then the analysis is not necessary because the animations have already been built with specific parameters (such as speed, angle of movement and distance between feet) which are taken into consideration when planning.

The first group works with root velocities and forces or rules working on a continuous space, or displacements within a grid. Different models include social forces [HFV00], rule-based models [Rey87], cellular automata [TLCDC01], flow tiles [Che04], roadmaps [SAC*07], continuum dynamics [TCP06], local fields [KSHF09], hybrid methods [SKH*11], and forces models parameterized by psychological and geometrical rules [PAB07]. They can easily represent agents by discs or cylinders to illustrate their steering behavior, but do not care about a final representation using 3D animated characters, so the output trajectory needs to be used to synthesize an animation following it. Synthesizing the animation from a small database can cause artifacts such as foot-sliding that need additional work to be removed [PSB11].

The second group works directly with the set of available animations to construct motion graphs [KGP08, ZS09, RZS10, MC12], or precomputed search trees [LK06]. These approaches try to reach the goal by connecting motions to each other [WP95], sometimes limiting the movements of the agents. Other methods try to use motion graphs in the first group combining it with path planners [vBEG11]. Having a large animation database reduces the limitations in terms of freedom of movement, but also makes the planning more time consuming. The ideal solution would be one that could find a good trade-off between these two goals: freedom of movement and fast planning.

Some approaches have tried to change the simulation paradigm by using more complex agent representations, such as footsteps. They can be physically based but generated off-line [FM12]. Or they can be generated online from an input path computed by a path planner [EvB10], or planning them using an inverse pendulum model instead of root positions [SKRF11]. Recent work [KBG*13] proposes the use of multiple domains of control focusing searches in more complex domains, only when necessary. The resulting behavior offers better results giving characters a better interactivity with the environment and other agents, but they fall in the first group of our classification since they do not take animation into account and need another process to synthesize it.

Some locomotion controllers are able to synthesize in real-time animations according to velocity and orientation parameters [TLP07]. Other locomotion controllers can accurately follow a footstep trajectory by extracting and parameterizing the steps of a motion capture database [vBPE10]. However they all need a very large database and their computational time does not allow to have many characters in real-time.

Our work belongs to the second group of the classification, since it uses an animation-based path planner. However instead of pre-computing a search tree with a few handmade animation clips, we pre-process motion capture data (which allows us to have more natural looking animations and larger

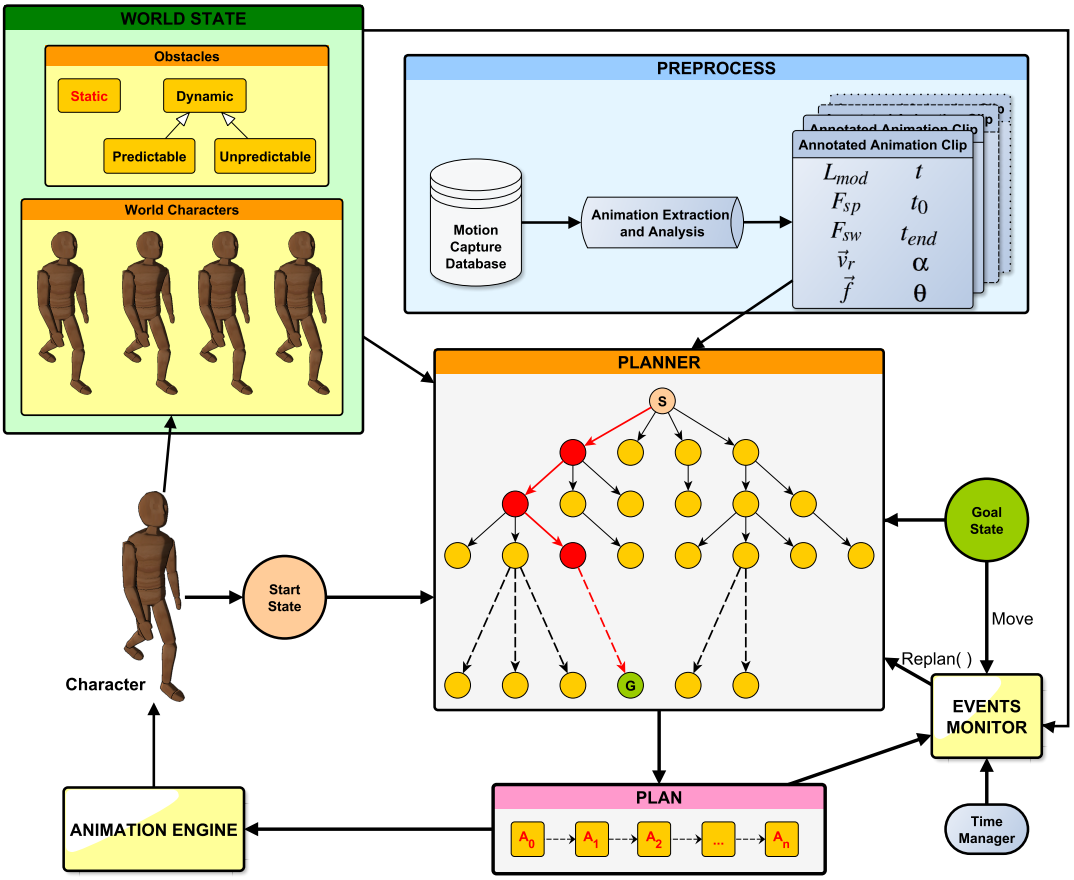


Figure 2: *Diagram showing the process required for the dynamic footstep planning algorithm*

variety), and extract actions from the input animations to compute a graph on the fly with an intelligent pruning based on logical transitions and a collision prediction system. Collisions are predicted and avoided for both static and deterministic dynamic obstacles, as well as for other agents since we expose all known trajectories.

3. Overview

Figure 2 illustrates the process of dynamic footstep planning for each character in real-time. The framework iterates over all characters in the simulation to calculate each individual foot step trajectory considering obstacles in the environment as well as other agents' calculated trajectories.

The *Preprocess* phase is responsible for extracting annotated animation clips from a motion capture database. The real-time *Planner* uses the annotated animations as transitions between state nodes in order to perform a path planning task to go from an input *Start State* to a *Goal State*. The output of the planner is a *Plan* consisting of a sequence of actions $A_0, A_1, ..., A_n$, which are clips that the *Animation Engine* must play in order to move the *Character* along the computed path. Both state and plan of the *Character* are then input to the *World State* and thus exposed to other agents' planners, together with the nearby static or dynamic obstacles. The *World State* is used to prune and accelerate the search in order to predict and avoid potential collisions. The *Time Manager* is responsible for checking the elapsed time between frames to keep track of the expiration time of the current plan. Finally the *Events Monitor* is in charge of detecting events that will force the planner to recompute a new path. The *Events Monitor* receives information from the *World State*, the *Time Manager*, *Goal State* and the character's current *Plan*. Events include: a possible invalid plan or the detection of a new dynamic obstacle or the goal position changing.

3.1. Events Monitor

The events monitor is the module of the system in charge of deciding when a new path needs to be recomputed. Elements that will trigger an event are:

- *Goal state changed*: when the goal changes its position or a new goal is assigned for the current character.
- *New agent or deterministic dynamic obstacle nearby*: other agents or dynamic obstacles enter the surrounding area of our character. A new path needs to be calculated to take into account the potential collision.
- *Collision against non-deterministic obstacle*: sometimes an unpredictable dynamic obstacle could lead to a collision (for example: a dynamic obstacle moved by the user), so when the events monitor detects such situation it triggers an event in order to react to it.
- *Plan expiration*: a way to ensure that each agent is taking into account the latest plans of every other agent is to give every plan an expiration time and force re-planning if this is reached. A time manager helps monitoring this task, but instead of a time parameter this event can also be measured and launched by a maximum number of actions that we want to perform (play) before re-planning.

4. Preprocess

During an offline stage, we analyze a set or a database of animation clips in order to extract the actions that our planner will then use as transitions between states. Each action consists of a sequence of skeleton configurations that perform a single animation step at a time, i.e., starting with one foot on the floor, until the other foot (swing foot) is completely resting on the floor. Our preprocess should work with any animation clip, since we tried both handmade and motion capture clips (from the CMU database [CMU13]). After analyzing each animation clip, we calculate mirrored animations. Mirroring animations is done in order to have each analyzed animation clip with either feet starting on the floor. The output of this stage is a set of annotated animations that can be used by the planner and the animation engine. This set can be easily serialized and stored to be reused for all instances of the same character type (same skeleton and the same scale, otherwise even if they share animations these could produce displacements of different magnitudes), reducing both preprocess time and the global memory consumption.

4.1. Locomotion Modes

In order to give our characters a wider variety and agility of movements we define different locomotion modes that need to be treated differently. Each animation clip will be tagged with its locomotion mode. We thus have the following set of locomotion modes:

- Walking: these are the main actions that will be used by the planner and the agents since they represent the most common way to move. We therefore have a wide variety of walks going from very slow to fast and in different angles (not just forward and backwards).
- Running: these are going to be treated in the same way as the walking actions with an additional cost penalty (since running consumes more energy than walking). We have also noticed empirically that for running actions it is not necessary to have as many different displacement angles as for walking actions.
- Turns: turns are going to be clips of animation where the agent turns in place or with a very small root displacement. They are going to be defined by their turning angle and velocity.
- Platform Actions: in this group we will find actions like jumping or crouching in order to avoid some obstacles. Such actions should have a high energy cost and should only be used in case of an imminent danger of collision.

While turns and platform actions need to be performed completely from start to end, and they do not have any intrinsic pattern we can easily detect, walking and running animations can be segmented by clips containing a single step. So animations of both walking and running locomotion modes will have a special treatment as we will need to extract the footsteps and keep only the frames of the animation covering a single step.

4.2. Footsteps Extraction

As previously mentioned in the paper, an action starts with one foot on the floor and ends when the other foot is planted on the floor. But animation clips, especially motion capture animations, do not always start and end in this very specific way. Therefore we need a foot plant extraction process to determine the beginning and end ending of each animation clip that will be used as an action.

Simply checking for the height of the feet in the motion capture data is not enough, since it usually contains noise and artifacts due to targeting. In most cases, when swinging the foot forwards while walking, the foot can come very close to the ground, or even traverse it.

Other techniques also incorporate the velocity of the foot during foot plant, which should be small. However this solution can also fail, since foot skating can introduce a large velocity. We detect foot plants using a height and velocity based detector similar to the method described in [vBE09], where foot plant detection is based on both height and time. First, the height-based test provides a set of foot plants, but only those where the foot plant occurs in a group of adjacent frames, are kept.

Our method combines this idea with changes on velocity for more accurate results, so we detect a foot plant when for a discretisized set of frames the foot is close to the ground for a few adjacent frames and with a change in velocity (deceleration, followed by being still for a few frames, and finishing with an acceleration). Notice that this method works for any kind of locomotion ranging from slow walking to running including turns in any direction.

4.3. Clip annotation

An analysis is performed by computing some variables over the whole duration of the animation. Each analyzed animation clip is annotated with the following information:

L_{mod}	Locomotion mode
F_{sp}	Supporting Foot
F_{sw}	Swing Foot
$\vec{v}_r$	Root velocity vector
$\vec{f}$	Foot displacement
t	Time duration
t_0	Initial time
t_{end}	End time
α	Movement angle
θ	Rotation angle
$\mathbb{P}$	Set of Sampled positions

Table 1: *Information stored in each annotated animation clip.*

Locomotion mode, indicates the type of animation (walk short step, walk long step, run, walk jump, climb, turn, etc). *Supporting foot* is the foot that is initially in contact with the floor, and the *swing foot* corresponds to the foot that is moving in the air towards the next footstep. The *supporting foot* is calculated automatically based on its height and velocity vector from frame to frame.

The *root velocity vector* indicates, taking the starting frame of the extracted clip as reference, the total local displacement vector of the root during the whole step. We therefore know the magnitude, the speed in m/s and the angle of its movement. Similarly, *foot displacement* tracks the movement of the swing foot.

Movement angle in degrees indicates the angle between the swing foot displacement vector and the initial root orientation. Therefore an angle equal to 0 means an action moving forward and 180 means it is a backward action. An Angle equal to 90 means an action moving to the left if the swing foot is the left one, or the right if the swing foot is the right one. Finally the *rotation angle* is the angle between the root orientation vector in the first and last frame of the clip.

t indicates the total time duration of the extracted clip, with t_0 and t_{end} storing the start and end point of the original animation that the extracted clip covers. These values will be used by the animation engine to play the extracted clip.

$\mathbb{P}$ corresponds to a set of sampled positions for certain joints of the character within an animation clip, and it is used for collision detection (see section 5.5)

5. Planning Footstep Trajectories

In this section, we first present the high level path planning on the navigation mesh. Then we define the problem domain we are dealing with when planning footsteps trajectories. Next we give details of the real-time search algorithm that we use as well as the pruning carried out to accelerate the search. Finally we explain how the collision detection and prediction is performed.

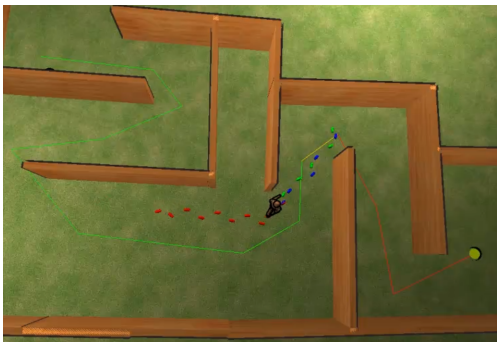


Figure 3: *High level path with local footstep trajectory between consecutive visible waypoints.*

5.1. High Level Path Planning

Footstep trajectories are calculated between waypoints of the high level path (see Figure 3). This path is calculated over the navigation mesh using Recast [Mon09]. An A* algorithm is used to compute the high level path, and then footstep trajectories are calculated between consecutive visible waypoints. So given a sequence of waypoints $\{w_i, w_{i+1}, w_{i+2}, ..., w_{i+n}\}$), if there is a collision-free straight line between w_i and w_{i+n}, then the footstep trajectory is calculated between those two waypoints, and any other intermediate point is ignored. This provides more natural trajectories as it avoids zig-zagging over unnecessary waypoints. Waypoints are considered by the planner as goal states, and each time that we change a waypoint the change of goal is detected by the events monitor, thus forcing a new path to be computed.

5.2. Problem Definition

The algorithm for planning footstep trajectories needs to calculate the sequence of actions that each agent needs to follow in order to go from their start position to their goal position. This means solving the problem of moving in a footstep domain between two given positions in a specific amount of time. Therefore, characters calculate the best trajectory based on their current state, the cost of moving to their destination and a given heuristic. The cost associated with each action is given by the bio-mechanical effort required to move (i.e: walking has a smaller cost than running, stopping for a few seconds may have a lower cost than wandering around a moving obstacle). The problem domain that we are dealing with is thus defined as:

$$\Omega = \left(\mathbb{S}, \mathbb{A}, c\left(s, s'\right), h(s, s_{goal})\right)$$

Where $\mathbb{S}$ is the state space and is defined as the set of states composed of the character's own state *self*, the world composition *environment*, and the *other agents* state. The action space $\mathbb{A}$ indicates the set of possible transitions in the state space and thus will have an impact on the branching factor of the planner. Each transition is an action, so we will have as many transitions as extracted clips times the possible speed variations we allow to introduce (we can for example reproduce a clip at half speed to obtain its displacement two times slower). Actions are then going to be defined by their corresponding annotated animation. $c\left(s,s'\right)$ is the cost associated with moving from state s to state s'. Finally $h(s,s_{goal})$ is the heuristic function estimating the cost to go from s to s_{goal}.

5.3. Real-Time Planning Algorithm

Planning footsteps trajectories in real time requires finding a solution in the problem domain Ω described earlier. The planner solution consists of a sequence $A_0,A_1,...,A_n$ of actions. Our planner interleaves planning with execution, because we want to be able to replan while consuming (playing) the action. For this purpose, we use a best-first search technique (e.g., A*) in the footstep problem domain, defined as follows:

- $\mathbb{S}$: the state space will be composed of the character's own state (defined by position, velocity, and the collision model chosen), the state of the other agents plus their plan, and the state and trajectory of the deterministic dynamic obstacles. For more details about collision models and obstacles avoidance see section 5.5.
- $\mathbb{A}$: the action space will consist of every possible action that can be concatenated with the current one without leading to a collision, so before adding an action we will perform all necessary collision checks.
- $c\left(s,s'\right)$: the cost of going from one state to another will be given by the energy effort necessary to perform the animation:

$$c\left(s,s'\right) = M\int_{t=0}^{t=T} e_s + e_w\left|v\right|^2 dt$$

  where M is the agent mass, T is the total time of the animation or action being calculated, v the speed of the agent in the animation, and e_s and e_w are per agent constants (for an average human, $e_s = 2.23\frac{J}{Kg.s}$ and $e_w = 1.26\frac{J.s}{Kg.m^2}$) [KWRF11].
- $h(s,s_{goal})$: the heuristic to reach the goal comes from the optimal effort formulation:

$$h(s,s_{goal}) = 2Mc^{opt}(s,s_{goal})\sqrt{e_s e_w}$$

  where $c^{opt}(s,s_{goal})$ is the cost of the optimal path to go from s to s_{goal}, in our case we chose the euclidian distance between s and s_{goal} [KWRF11]. The optimal effort for an agent in a scenario is defined as the energy consumed in taking the optimal route to the target while traveling at the average walking speed: $v_{av} = \sqrt{\frac{e_s}{e_w}} = 1.33m/s$

Taking all these components into consideration the planner can search for the path with least cost and output the footstep position with their time marks that the animation engine will follow by playing the sequence of actions planned (see figure 4).

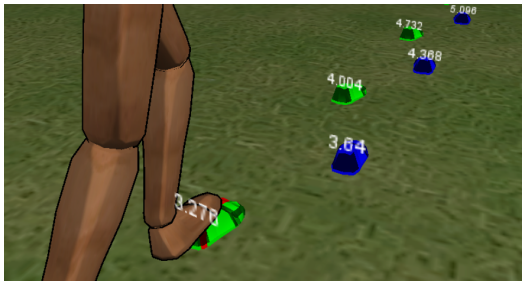


Figure 4: *Footsteps trajectory with time constraints that need to be followed by the animation controller.*

5.4. Pruning Rules

In order to accelerate the search we can add simple rules to help prune the tree and reduce the branching factor. A straight forward way to halve the size of the tree consists of considering only consecutive actions starting with the opposite foot. So given a current node with a supporting foot, expand the node only for transitions that have that same foot as the swing foot. Actions which are not possible due to locomotion constraints on speed or rate of turning are also pruned to ensure natural character motion (so after a staying still animation, we will not allow a fast running animation). The next pruning applied is based on collision prediction as we will see in the following section. The idea is that when a node is expanded and a collision is detected, the whole graph that could be expanded from it gets automatically pruned. The pruning process reduces the branching factor of the search, and also ensures natural footstep selection

5.5. Collision Prediction

While expanding nodes the planning algorithm must check for each expanded node whether the future state is collision free or not. If it is collision free, then it maintains that node and continues expanding it. Otherwise, it will be discarded. In order to have large simulations in complex environments we need to perform this pruning process in a very fast manner.

In order to predict collisions against other agents or obstacles (both dynamic or static), we introduce a multi-resolution collision detection scheme which performs collision checks for two resolution levels. Our lowest resolution collision detection model is a simple cylinder centered at the root of the agent with a fixed radius. The higher resolution model consists of five cylinders around the end joints (head, hands and feet) that are used to make finer collision tests Figure 5.

We could introduce more collision models, where high resolution ones will be executed only in case of detecting collisions using the coarser ones. At the highest complexity mode we could have the full mesh collision check, but for the purpose of our simulation the 5 cylinders model gives us enough precision to avoid agents walking with their arms intersecting against other agents as they swing back and forth. Compared against simpler approaches that only consider obstacle detection against a cylinder, our method gives better results since it allows us to have closer interactions between agents. All obstacles have simple colliders (boxes, spheres, capsules) to accelerate the collision checks by using a fast physics ray casting test.

It is also important to mention that collision tests are not only performed using the initial and end positions of the expanded node, but also with sub-sampled positions inside the animation (for the 5 cylinder positions). For example, an agent facing a thin wall as a start position and the other side of the wall as end position of its current walk forward step. If we only check for possible collisions with those start and end positions we would not detect that the agent is actually going through the wall.

The sub-sample for each animation is performed off-line and stored in the annotated animation. To save memory, this sampling is performed at low frequencies and then in real time intermediate positions can be estimated by linear interpolation.

Finally, we provide the characters with a surrounding view area to maintain a list of obstacles and agents that are potential threats to our path (see figure 6). For each agent, we are only interested in those obstacles/agents that fall within the view area in order to avoid running unnecessary collision tests.

5.5.1. Static World

Static obstacles are part of the same static world that is used to compute the navigation mesh with Recast [Mon09]. They do not need to have a special treatment since the high-level path produces waypoints that avoid collisions with static obstacles..

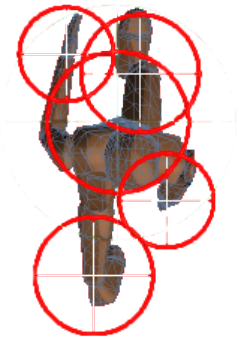


Figure 5: *Collision model of 5 cylinders around the head, the left and right hands, and the left and right foot.*

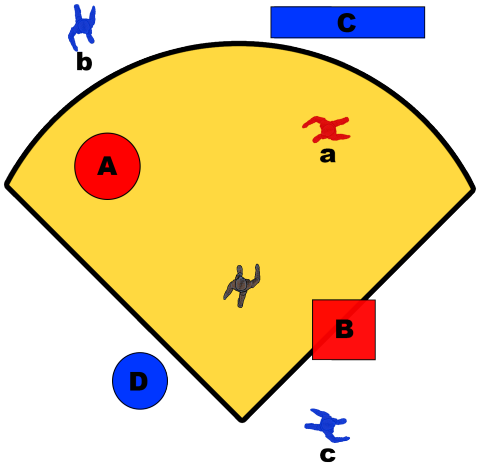


Figure 6: *When planning we only consider obstacles and agents that are inside the view area. Obstacles A, B and agent a are inside it and the agent will try to avoid them, while it will ignore obstacles C, D and agents b and c .*

5.5.2. Deterministic Dynamic Obstacles and Other Agents

Deterministic obstacles move with a predefined trajectory. Other agents have precomputed paths which can be queried to predict their future state. To avoid interfering with those paths we allow access to their temporal trajectories. So, for each expanded node with state time t we check for collisions with every obstacle and agent that falls inside his view area at their trajectory positions at time t. Figure 7 shows an example of an agent avoiding two dynamic obstacles.

5.5.3. Unpredictable Dynamic Obstacles

Unlike deterministic dynamic obstacles and other agents, unpredictable dynamic obstacles are impossible to be accounted for while planning. Therefore they can be ignored when expanding nodes, but we need a fast way to react to them. This is the reason why we need the events monitor to detect immediate collisions and force re-planning. Figure 8 shows an example where a wall is arbitrarily moved by the user and the agent needs to continuously re-plan its trajectory.

6. Animation Engine

The animation engine is in charge of playing the output sequence of actions given by the planner. These actions contain all the data in the annotated animation. When a new action is

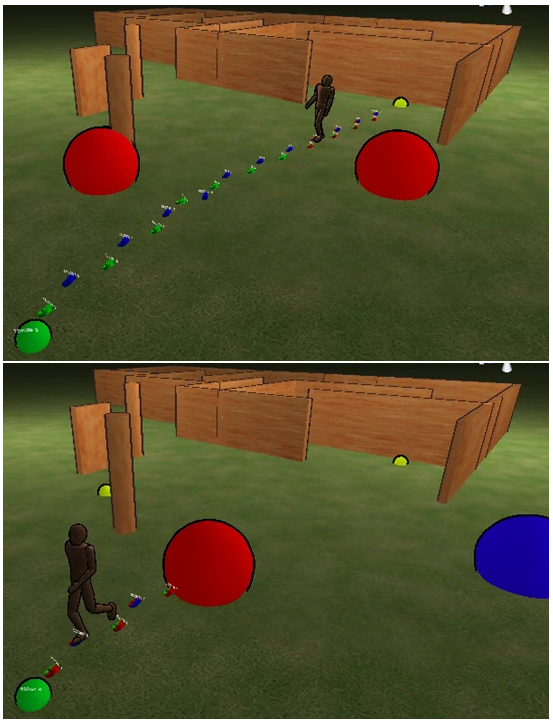


Figure 7: *An agent planning with two dynamic obstacles in front of him (top). After executing some steps the path is re-planned. The blue obstacle indicates that it is not in his nearby area anymore, so that obstacle is not considered in the collision check of this new plan. (bottom)*

played it sets t_0 as the initial time of the animation. When the current animation reaches t_{end} the animation engine blends the current animation with the next one in the queue.

The Animation Engine also tracks the global root position and orientation, and applied rotation corrections by rotating the whole character using the rotation values of the annotated animation (rotation angle θ). The blending time between actions can be user defined within a short time (for example 0.5s).

7. Results

The presented framework has been implemented using the ADAPT simulation platform [SMKB13] which works with Unity Game Engine [Uni13] and C# scripts. Our current framework can simulate around 20 agents at approximately 59-164 frames per second (depends on the maximum planning time allowed), and 40 agents at 22-61 frames per second (INtel Core i7-2600k CPU @ 3.40GHz and 16GB RAM). Figure 9 shows the frame rates achieved on average for an increasing number of agents. The black line corresponds to a maximum planning time of 0.01s, and the red line corresponds to 0.05s. Additionally, by setting planner

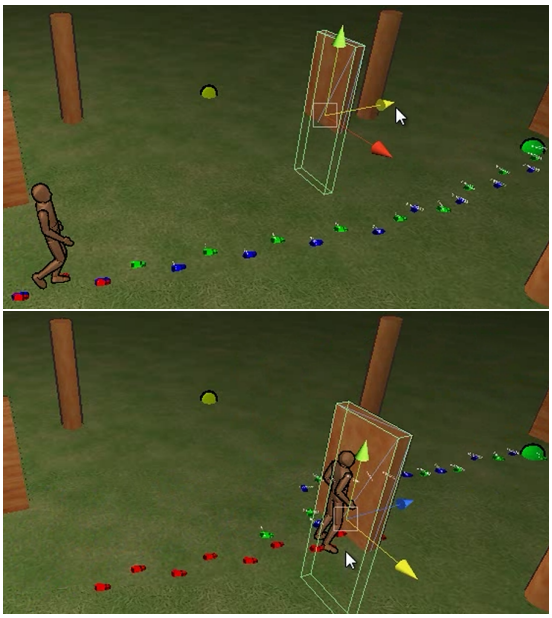


Figure 8: *An agent reacting to a non-deterministic obstacle by re-planning his path.*

parameters such as the horizon of the search, we can achieve significant speedup at the expense of solution fidelity. For example, we can produce purely reactive simulations where the character only plans one footstep ahead by reducing the search horizon to 1.

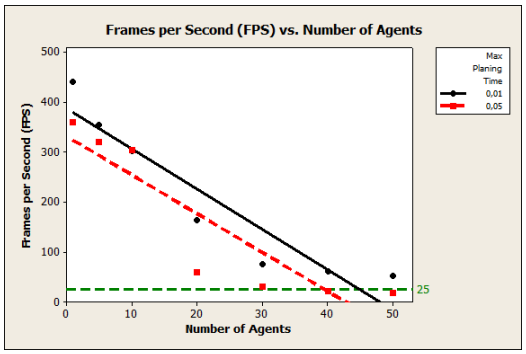


Figure 9: *This graph shows the frames per second on average for different simulations with increasing number of agents. We have used two values for the maximum planning time: 0.01 resulting in higher frame rates, and 0.05 resulting in lower frame rates but better quality paths*

The results showed have been made with a database of 28 motion captured animations. This is a small number compared to approaches based on motion graphs (generally hav-

ing around 400 animation clips), but a large number compared with techniques based on handmade animation (such as pre-computed search trees). This decision allows us to achieve results that look natural and yet can be used for real time applications.

Our approach solves different scenarios where several agents are simulated in real-time achieving natural looking paths while avoiding other obstacles and characters (see accompanying videos). The quality of the results in terms of natural paths and collision avoidance depends on the planner. The planner will be given a specific amount of time to find a solution (which translates in how many nodes of the graph are expanded). Obviously when we allow larger search times (larger number of nodes to expand) the resulting trajectory looks more natural and is collision free, but at the expense of being more computationally expensive. Alternatively, if we drastically reduce the search time (smaller number of nodes to expand) we may end up having collisions as we can see in the resulting videos and in Figure 10.

Interleaving planning with execution provides smooth animations, since not all the characters plan their paths simulateneously. At any time, the new plan is calculated with the start position being the end position of the current action.

We have also shown how the Events Monitor can successfully plan routes when deterministic obstacle invalidate a character's plan, as well as efficiently react to non deterministic obstacles (see Figures 7 and 8)

8. Conclusions and Future Work

We have presented a multi-agent simulation approach where planning is done in the action space of available animations. Animation clips are analyzed and actions are extracted and annotated, in order to be used in real time to expand a search tree. Nodes are only expanded if they are collision free. To predict collisions we sample animations and use a new collision model with colliders for each end joint (head, hands and feet). This way we are able to simulate agents avoiding more detailed collisions. The presented framework handles both deterministic and non-deterministic obstacles, since the former can be taken into consideration when planning, while the later needs a completely reactive behavior.

Unlike pre-computed search trees our set of transitions is composed of actions, and mainly footsteps, which allows us to build online the search tree and to dynamically prune it, considering not only start and goal positions, but also departure and arrival times. An events monitor can help us to decide when to re-plan the path, based on the environment situation such as obstacle proximity or velocity.

We would like to further extend the hierarchical nature of this work to add granularity (both in models and domains) to adaptively switch between them [KCS10, Lac02, SG10]. Solutions from a coarser domain could also be reused to accelerate the search into a finer domain, using techniques such as tunneling [GCB*11]. Another idea would be to have a special class of actions constituting a reactive domain that would only be used in case of an imminent threat. Since nondeterministc obstacles invalidating the current plan force to replan constantly, it would be interesting to carry out a quantitative study on the impact of the number of nondeterministic obstacles in the frame rate obtained for different number of agents.

As Illustrated in 9, the computational complexity of our framework scales linearly with number of agents. By reducing the search depth and maximum planning time, we can simulate a larger crowd of characters at interactive rates. Choosing the optimal value of these parameters that balance computational speed and agent behavior is an interesting research direction, and the subject of future work. Our framework is not memory bound, and is amenable to parallelization with each agent planning on an independent thread.

Notice that memory is required per animation (to store sub-sampled animations) and not per agent in the simulation, therefore increasing the size of the simulated group of agents would not have an impact on the memory requirements of our system. If we wanted to simulate crowds of characters we would need more CPU power, but not memory as long as we had more instances of characters sharing the same skeleton and animations.

We would also like to improve our base search algorithm with a faster one taking into account repairing capacities such as ARA* [LGT03]. Having more characters and different sets of actions that can be used depending on the situation, like a reaction domain, would also accelerate the search and give better results to our simulations in constantly changing dynamic virtual environments.

Acknowledgements

This work has been partially funded by the Spanish Ministry of Science and Innovation under Grant TIN2010-20590-C01-01. A. Beacco is also supported by the grant FPUAP2009-2195 (Spanish Ministry of Education). We would also like to acknowledge Francisco Garcia for his implementation of the Best First Search algorithm.

References

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

[CMU13] CMU: Cmu graphics lab motion capture database, 2013. http://mocap.cs.cmu.edu/. 4

[EvB10] EGGES A., VAN BASTEN B.: One step at a time: Animating virtual characters based on foot placement. *The Visual Computer 26*, 6-8 (apr 2010), 497–503. 2

[FM12] FELIS M., MOMBAUR K.: Using Optimal Control Methods to Generate Human Walking Motions. *Motion in Games* (2012), 197–207. 2

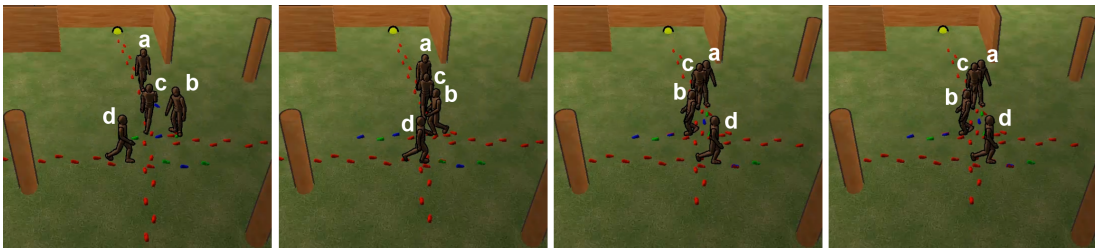


Figure 10: *Example with four agents crossing paths with a drastically reduced search time resulting in agents **a** and **c** not being able to avoid intersection as seen in the last two images of this secuence. Also notice how agent **b**, walks straight towards **c**, steps back and then continues, instead of following a smooth curve around **c**.*

[GCB*11] GOCHEV K., COHEN B., BUTZKE J., SAFONOVA A., LIKHACHEV M.: Path planning with adaptive dimensionality. In *Fourth Annual Symposium on Combinatorial Search* (2011). 9

[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 2013 ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation* (2013), SCA. 2

[KCS10] KRING A. W., CHAMPANDARD A. J., SAMARIN N.: Dhpa* and shpa*: Efficient hierarchical pathfinding in dynamic and static game worlds. In *Sixth Artificial Intelligence and Interactive Digital Entertainment Conference* (2010). 9

[KGP08] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. In *ACM SIGGRAPH 2008 classes* (2008), ACM, p. 51. 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. 2

[KWRF11] KAPADIA M., WANG M., REINMAN G., FALOUTSOS P.: Improved benchmarking for steering algorithms. In *Motion in Games*. Springer, 2011, pp. 266–277. 6

[Lac02] LACAZE A.: Hierarchical planning algorithms. In *AeroSense 2002* (2002), International Society for Optics and Photonics, pp. 320–331. 9

[LGT03] LIKHACHEV M., GORDON G., THRUN S.: Ara*: Anytime a* with provable bounds on sub-optimality. *Advances in Neural Information Processing Systems (NIPS) 16* (2003). 9

[LK06] LAU M., KUFFNER J. J.: Precomputed search trees: planning for interactive goal-driven animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2006), Eurographics Association, pp. 299–308. 2

[MC12] MIN J., CHAI J.: Motion Graphs++. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 1. 2

[Mon09] MONONEN M.: Recast navigation toolkit webpage, 2009. http://code.google.com/p/recastnavigation/. 1, 5, 7

[OP13] OLIVA R., PELECHANO N.: Neogen: Near optimal generator of navigation meshes for 3d multi-layered environments. *Computer & Graphics 37*, 5 (2013), 403–412. 1

[PAB07] PELECHANO N., ALLBECK J. M., BADLER N. I.: Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 99–108. 2

[PSB11] PELECHANO N., SPANLANG B., BEACCO A.: Avatar locomotion in crowd simulation. In *International Conference on Computer Animation and Social Agents (CASA)* (Chengdu, China, 2011), vol. 10, pp. 13–19. 2

[Rey87] REYNOLDS C. W.: Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH Computer Graphics* (1987), vol. 21, ACM, pp. 25–34. 2

[RZS10] REN C., ZHAO L., SAFONOVA A.: Human Motion Synthesis with Optimization-Based Graphs. *Computer Graphics Forum (Proceedings of Eurographics 2010) 29*, 2 (2010). 2

[SAC*07] SUD A., ANDERSEN E., CURTIS S., LIN M., MANOCHA D.: Real-time path planning for virtual agents in dynamic environments. In *Virtual Reality Conference, 2007. VR'07. IEEE* (2007), IEEE, pp. 91–98. 2

[SG10] STURTEVANT N. R., GEISBERGER R.: A comparison of high-level approaches for speeding up pathfinding. *Artificial Intelligence and Interactive Digital Entertainment (AIIDE)* (2010), 76–82. 9

[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. 2

[SKRF11] SINGH S., KAPADIA M., REINMAN G., FALOUTSOS P.: Footstep Navigation for Dynamic Crowds. *Computer Animation And Virtual Worlds 22*, April (2011), 151–158. 1, 2

[SMKB13] SHOULSON A., MARSHAK N., KAPADIA M., BADLER N. I.: Adapt: the agent development and prototyping testbed. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2013), I3D '13, ACM, pp. 9–18. 8

[TCP06] TREUILLE A., COOPER S., POPOVIĆ Z.: Continuum crowds. In *ACM Transactions on Graphics (TOG)* (2006), vol. 25, ACM, pp. 1160–1168. 2

[TLCDC01] TECCHIA F., LOSCOS C., CONROY-DALTON R., CHRYSANTHOU Y.: Agent behaviour simulator (abs): A platform for urban behaviour development. 2

[TLP07] TREUILLE A., LEE Y., POPOVIĆ Z.: Near-optimal character animation with continuous control. In *ACM Transactions on Graphics (TOG)* (2007), vol. 26, ACM, p. 7. 2

[Uni13] UNITY: Unity - game engine, 2013. http://unity3d.com/. 8

[vBE09] VAN BASTEN B. J. H., EGGES A.: Evaluating distance metrics for animation blending. In *Proceedings of the 4th International Conference on Foundations of Digital Games* (New York, NY, USA, 2009), FDG '09, ACM, pp. 199–206. 4

[vBEG11] VAN BASTEN B., EGGES A., GERAERTS R.: Combining Path Planners and Motion Graphs. *Computer Animation and Virtual Worlds 22*, 1 (2011), 59–78. 2

[vBPE10] VAN BASTEN B. J. H., PEETERS P. W. A. M., EGGES A.: The step space: example-based footprint-driven motion synthesis. *Computer Animation and Virtual Worlds 21*, 3-4 (May 2010), 433–441. 2

[WP95] WITKIN A., POPOVIC Z.: Motion warping. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM, pp. 105–108. 2

[ZS09] ZHAO L., SAFONOVA A.: Achieving good connectivity in motion graphs. *Graphical Models 71*, 4 (2009), 139–152. 2

Footstep Parameterized Motion Blending using Barycentric Coordinates

A. Beacco[1], N. Pelechano[1], M. Kapadia[2] & N.I. Badler[3]

[1]Universitat Politècnica de Catalunya
[2]Disney Research Zurich
[3]University of Pennsylvania

Abstract

This paper presents a real-time animation system for fully-embodied virtual humans that satisfies accurate foot placement constraints for different human walking and running styles. Our method offers a fine balance between motion fidelity and character control, and can efficiently animate over sixty agents in real time (25 FPS) and over a hundred characters at 13 FPS. Given a point cloud of reachable support foot configurations extracted from the set of available animation clips, we compute the Delaunay triangulation. At runtime, the triangulation is queried to obtain the simplex containing the next footstep, which is used to compute the barycentric blending weights of the animation clips. Our method synthesizes animations to accurately follow footsteps, and a simple IK solver adjusts small offsets, foot orientation, and handles uneven terrain. To incorporate root velocity fidelity, the method is further extended to include the parametric space of root movement and combine it with footstep based interpolation. The presented method is evaluated on a variety of test cases and error measurements are calculated to offer a quantitative analysis of the results achieved.

Keywords:
Character animation, Crowd simulation, Footsteps controller

1. Introduction

Crowd simulation research has matured in recent years with important applications in training, building design, psychological studies, and video-games. All these applications benefit from having fully-embodied virtual human characters animated in real-time while accurately satisfying control objectives without any noticeable artifacts.

Algorithms that generate center of mass (COM) trajectories [1, 2, 3, 4] lead to ambiguities when trying to superimpose a fully articulated virtual human to follow them, thus producing foot-sliding artifacts when no suitable animation is found, or when the root orientation and the displacement vector of the animation do not match. Different animations can be blended by tweaking some of the upper body joints [5] to minimize artifacts, at the expense of constant updates to account for the decoupling between the crowd simulation and the animation system.

Footstep-based control systems [6, 7] output a list of space-time foot-plants to define a fine-grained trajectory with fewer ambiguities that can solve more complex scenarios (e.g., complex manipulation tasks requiring careful control of the lower body, or collaborative tasks, such as careful sidestepping to make way for another agent in a narrow corridor). To realistically represent such simulations, we need a method to synthesize animations that accurately follow the output trajectory, i.e., accurate placement of feet with space-time constraints. This problem is traditionally known as the *stepping stone* problem. Moreover, the output trajectory can be modified by external perturbations such as uneven terrain.

We present an online animation synthesis technique for fully embodied virtual humans that satisfies foot placement constraints for a large variety of locomotion speeds and styles (see Fig. 1). Given a database of motion clips, we precompute multiple parametric spaces based on the motion of the root and the feet. A root parametric space is used to compute a weight for each available animation based on root velocity. Two foot parametric spaces are based on a Delaunay triangulation of the graph of possible foot landing positions. For each foot parametric space, blending weights are calculated as the barycentric coordinates of the next footstep position for the triangle in the graph that contains it. These weights are used for synthesizing animations that accurately follow the footstep trajectory while respecting the singularities of the different walking styles captured.

Blending weights calculated as barycentric coordinates are used to reach the desired foot landing by interpolating between several proximal animations, and IK is used to adjust the final position of the support foot to correct for minor offsets, foot step orientation and the angle of the underlying floor.

Since foot parametric space only considers final landing positions of the feet without taking into account root velocity, this may lead to the selection of animations that satisfy position constraints but introduce discontinuities in root velocity. To incorporate root velocity fidelity we present a method that can integrate both foot positioning and root velocity fidelity. Our method also allows the system to recover nicely when the input

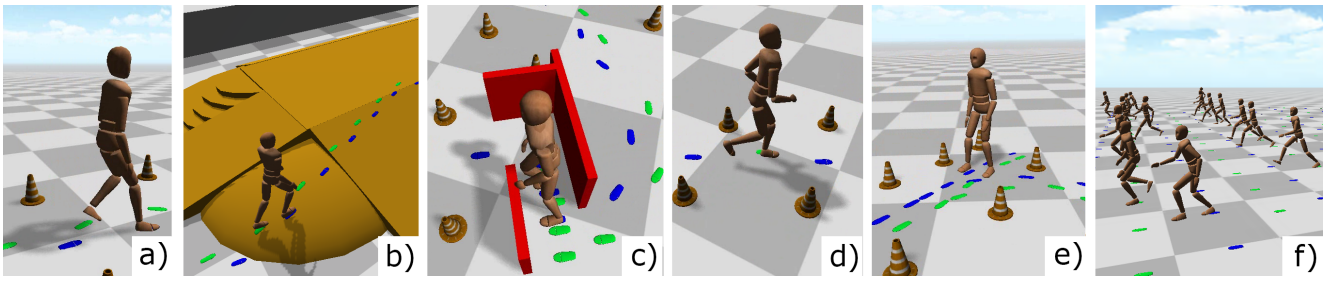


Figure 1: An autonomous virtual human navigating a challenging obstacle course (a), walking over a slope (b), exercising careful foot placement constraints including side-stepping (c), speed variations (d), and stepping back (e). The system can handle multiple agents in real time (f).

foot trajectory contains steps that are not possible to perform with the given set on animations (for example, due to extreme distance between steps).

The presented method is evaluated on a variety of test cases and error measurements are calculated to offer a quantitative analysis of the results achieved. Our framework can efficiently animate over sixty agents in real time (25 FPS) and over a hundred characters at 13 FPS, without compromising motion fidelity or character control, and can be easily integrated into existing crowd simulation packages. We also provide the user with control over the trade-off between footstep accuracy and root velocity.

2. Related Work

Locomotion synthesis can be tackled from different points of view depending on how the character is being controlled. If a user controls the character with a 3rd person controller, it is common to work on a root velocity basis, because the user wants to move the character around in an agile way. In such cases, like video-games, real-time response is critical and artifacts such as foot skating can be ignored. Optimization based approaches [8] are able to synthesize animations that conform to velocity and orientation constraints. However, they need a very large database and their computational time does not allow many characters in real-time. Semi-procedural animation systems [9] work with a small set of animations and use inverse kinematics only over the legs to ensure ground contact and to adapt the feet to possible slopes of the terrain, but they are unable to follow footstep trajectories.

Animation systems for autonomous agents must be computationally efficient to animate a multitude of characters in real-time, and need to follow different control trajectories, depending on the controller used. Controllers that account for animation constraints while computing control decisions such as motion graphs [10, 11, 12, 13] or precomputed search trees [14] can simply playback the animation sequence. These approaches try to reach the goal by connecting series of motion[15], which sometimes limits the movements of the agents. The main issues with motion graphs are that they require a very large amount of animation clips (over 400) and have a high computational cost which makes them not suitable for large groups of agents in real-time. Precomputed search trees can handle groups, but work with a few animation clips and are unable to synthesize new animations.

Approaches that ignore animation constraints produce center of mass trajectories for the animation system to follow. Different models include social forces [2], rule-based approaches [1], flow tiles [16], roadmaps [17], continuum dynamics [3], and force models parametrized by psychological and geometrical rules [4]. These techniques can easily simulate hundreds and thousands of characters in real-time, but do not account for locomotion constraints, thus producing artifacts such as foot-sliding which require correction and simulation updates [5].

Considering the root velocity as the input parameter for character control, numerous approaches can synthesize smooth, versatile and more plausible locomotion animations [18, 9]. Some approaches have also used the idea of selecting animations from a Delaunay triangulation of all the available animation clips [19, 20]. But all of these approaches are restricted to the root for performing character control.

There has been a recent surge in approaches that produce footstep trajectories for character control. They can be physically based but generated off-line [21], be generated online from an input path computed by a path planner [6], or use simplified control dynamics to produce bio-mechanically plausible footstep trajectories for crowds [7]. These approaches often show their animation results off-line using tools such as 3D Max [22].

Footstep-driven animation systems [23] produce unnatural results using procedural methods. The work in [24] uses a statistical dynamic model learned from motion capture data in addition to user-defined space-time constraints (such as footsteps) to solve a trajectory optimization problem. In [25] random samples of footsteps make a roadmap going from one point to another which is used to find a minimum-cost sequence of motions matching it and then retarget to the exact foot placements. The work in [26, 27] performs a global optimization over an extracted center of mass trajectory to maximize the physical plausibility and perceived comfort of the motion, in order to satisfy the footprint constraints. Recent solutions [6, 28, 29] adopt a greedy nearest-neighbor approach over larger motion databases. To ensure spatial constraints, the character is properly aligned with the footsteps and reinforced with inverse kinematics, while temporal constraints are satisfied using time warping. These techniques achieve highly accurate results in terms of foot positioning, but their computational cost

makes them unsuitable for real-time animation of large groups of agents.

Comparison to Prior Work. Our method produces visually appealing results with foot placement constraints, using only a handful of motion clips, and can seamlessly follow footstep-based control trajectories while preserving the global appearance of the motion. Compared to [9], we exploit the combination of multiple parameter spaces for footstep-precision control. This reduces the dimensionality of the problem, compared to [29]. Unlike previous work in the literature, our method can synthesize animations for a large number of characters in real time, following footstep trajectories for different walking styles and even running motions with a small flying phase.

3. Framework Overview

Animating characters in real time animations has different requirements depending on the application. In many applications, the user only wants to control the direction of movement and speed of the root, but there are other situations where a finer control of the foot positioning is required. For example, the user may want to respect different walking gaits depending on the terrain, to make the character step over stones to cross a river, or walk through some space full of holes whilst avoiding falling. For this purpose we have developed a framework to animate virtual characters following footstep trajectories, while still being able to follow trajectories based on the movement of the COM when necessary.

Online locomotion systems [9] traditionally produce synthesized motions that follow a COM trajectory, with procedural corrections for uneven terrain. These methods can nicely follow COM trajectories, but they lack control over the style of walking and the kind of steps. For instance, we cannot control whether in order to walk fast, the character will move with large distances between steps or with a fast sequence of short steps. This is the main issue we address in our work: to provide an animation system that is able to accurately follow footstep trajectories while meeting real-time constraints, and that can scale to handle large groups of animated characters .

For this purpose, we introduce two parametric spaces based on the position of each foot: Ω_{f_L} and Ω_{f_R}, and switch between the two depending on the swing foot, as well as a parametric space based on the root movement Ω_{f_R}. Our technique takes into account both displacement (from Ω_{f_L} and Ω_{f_R}) and speed (from Ω_r) to ensure the satisfaction of both spatial and temporal constraints. Our system provides the user with the flexibility to choose between different control granularities ranging from exact foot positioning to exact root velocity trajectories. Fig. 2 shows our framework.

4. Footstep-based Locomotion

The main goal of the Footstep-based Locomotion Controller is to accurately follow a footstep trajectory, i.e., to animate a fully articulated virtual human to step over a series of footplants with space and velocity constraints. The system must meet real-time constraints for a group of characters, should be robust enough to handle sparse motion clips, and needs to produce synthesized results that are void of artifacts such as foot sliding and collisions.

4.1. Motion Clip Analysis

From a collection of cyclic motion clips[1], we need to extract individual footsteps. Each motion clip contains two steps, one starting with the left foot on the floor, and one starting with the right foot. A step is defined as the action where one foot of the character starts to lift-off the ground, moves in the air and finishes when it is again planted on the floor. We say that a footstep corresponds to one foot when that foot is the one performing the action previously described. The foot that stays in contact with the floor for most of the duration of the footstep is called the supporting foot, since it supports the weight of the body. This applies even for running motions, where the support foot goes into fly mode for a short phase of the footstep, but it is still the one supporting the weight during most of the footstep.

During an offline analysis, each motion clip m_i is annotated with the following information: (1) $\mathbf{v}_i^r$: Root velocity vector. (2) $\mathbf{d}_i^L$: Displacement vector of the left foot. (3) $\mathbf{d}_i^R$: Displacement vector of the right foot.

Similar to [9], animations are analyzed in place, that is, we ignore the original root forward displacement, but keep the vertical and lateral deviations of the motion. This allows an automatic detection of foot events, such as lifting, landing or planting, from which we can deduce the displacement vector of each foot. For example, the displacement vector of the left foot $\mathbf{d}_i^L$ is obtained by subtracting the right foot position at the instant of time when the left foot lands, from the right foot position at the instant of time when the left foot is lifting off. These displacements will be later used to move the whole character, eliminating any foot sliding. By adding $\mathbf{d}_i^L$ to $\mathbf{d}_i^R$ and knowing the time duration of the clip, we can calculate the average root velocity vector $\mathbf{v}_i^r$ of the clip m_i.

This average velocity is used to classify and identify animations, by providing an example point which is the input for the polar gradient band interpolator (where each example point represents a velocity in a 2D parametric space). Gradient band interpolation specifies an influence function associated with each example, which creates gradient bands between the example point and each of the other example points. These influence functions are normalized to get the weight functions associated with each example. However the standard gradient band interpolation is not well suited for interpolation of examples based on velocities. The polar gradient band interpolation method is based on reasoning that in order to get more desirable behavior for the weight functions of example points that represent velocities, the space in which the interpolation takes place should take on some of the properties of a polar coordinate system. It allows for dealing with differences in direction

[1] Although cyclic animations are not strictly required by our method, they help find smoother transitions between consecutive footsteps and are preferred by most standard animation systems [9].

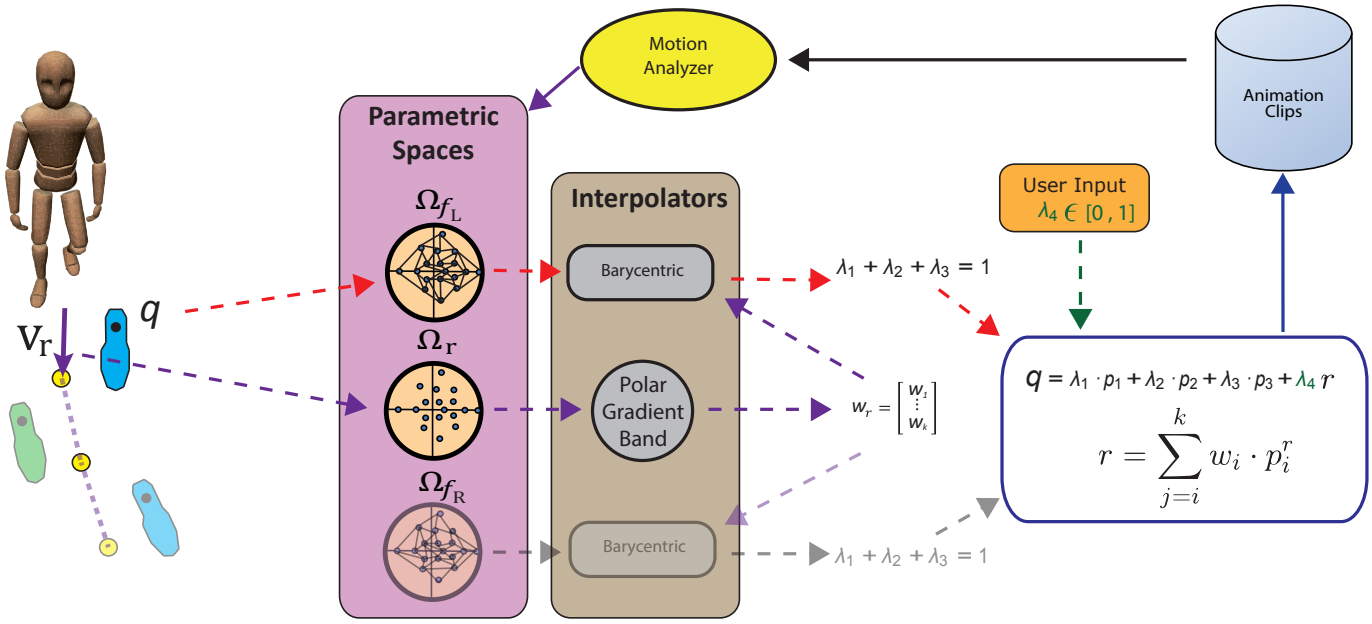


Figure 2: Online selection of the blend weights to accurately follow a footstep trajectory. Ω_r uses a gradient band polar based interpolator [9] to give a set of weights w_j, which are then used by the barycentric coordinates interpolator to tradeoff between footstep and COM accuracy.

and magnitude rather than differences in the Cartesian vector coordinate components. For more details we refer the reader to [9].

Each motion clip is then split into two animation steps A_i^L for the left foot and A_i^R for the right foot. For each foot, we need to calculate all the possible positions that can be reached based on the set of animation steps available. Since the same analysis is performed for both feet separately, from now on we will not differentiate between left and right for the ease of exposition. For each individual step animation A_i and given an initial root position, we want to extract the foot landing position p_i, if the corresponding section of its original clip was played. This is calculated by summing the root displacement during the section of the animation with the distance vector between the root projection over the floor and the foot position in the last frame.

The set $\{p_i | \forall i \in [1, n]\}$ where n is the number of step animations, provides a point cloud. Fig. 3 shows the Delaunay triangulation that is calculated for the point cloud of landing positions. This triangulation is queried in real time to determine the simplex that contains the next footstep in the input trajectory. Once the triangle is selected, we will use its three vertices p_1, p_2 and p_3 to compute the blending weights for each of the corresponding animations A_1, A_2 and A_3.

4.2. Footstep and Root Trajectories

Our system can work with both footstep trajectories and COM trajectories. A footstep trajectory will be given as an ordered list of space-time positions with orientations, whether it is precomputed or generated on-the-fly.

The input footstep trajectory may be accompanied by its associated root trajectory (a space-time curve, rather than a list of points, and an orientation curve), or else we can automatically compute it from the input footsteps by interpolation. This is

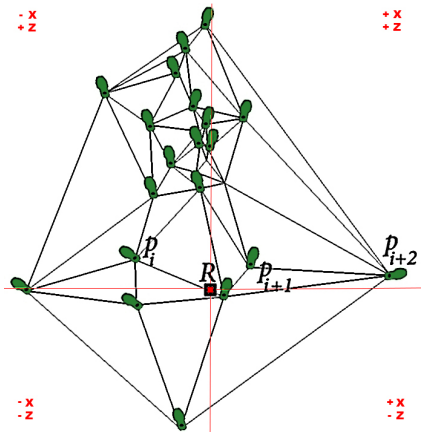


Figure 3: Delaunay triangulation for the vertices representing the landing positions (p_i, p_{i+1}, p_{i+2},...) of the left foot when the root, R is kept in place.

achieved by computing the projection of the root on the ground plane, as the midpoint of the line segment joining two consecutive footsteps. The root orientation is then computed as the average between the orientation vectors of each set of consecutive steps. This provides us with a sequence of root positions and orientations which can be interpolated to approximate the motion of the root over the course of the footstep trajectory.

4.3. Online Selection

During run time, the system animates the character towards the current target footstep. If the target is reached, the next footstep along the trajectory is chosen as the next target. For each footstep q_j in the input trajectory $\{q_1, q_2, q_3, ..., q_m\}$ we need to align the Delaunay triangulation graph with the current root position and orientation. Then the triangle containing the next foot

position is selected as the best match to calculate the weights required to nicely blend between the three animations in order to achieve a footstep that will land as close as possible to the desired destination position q_j (Fig. 4). Notice that these weights are applied equally to all the joints in the skeleton, which means that at this stage we cannot accurately adjust the specific foot orientation required by each footstep in the input trajectory.

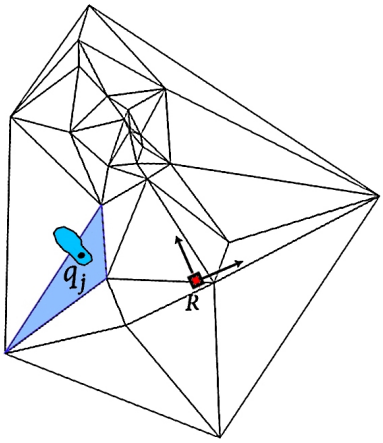


Figure 4: By matching root position and orientation, we can determine the triangle containing the destination position for the landing position q_j.

4.4. Interpolation

Footstep parameters change between successive footplants, remaining constant during the course of a single footstep (several frames of motion). Therefore we need to compute the best interpolation for each footstep, blend smoothly between consecutive steps, and apply the right transformation to the root in order to avoid foot-sliding or intersections with the ground.

To meet these requirements, we use a barycentric coordinates based interpolator in Ω_{f_L} and Ω_{f_R}, and constrain the solution based on the weights computed in Ω_r. This allows us to animate a character at the granularity of footsteps, while simultaneously accounting for the global motion of the full body.

If we only consider the footstep parametric space, then the vertices of the selected triangle are those that can provide the best match for the desired foot position. The barycentric coordinates of the desired footstep are calculated for the selected triangle as the coordinates that satisfy:

$$q_j = \lambda_1 \cdot p_1 + \lambda_2 \cdot p_2 + \lambda_3 \cdot p_3, \qquad (1)$$
$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

where p_1, p_2 and p_3 are the positions of the foot landing if we run animation steps A_1, A_2 and A_3 respectively. The calculated barycentric coordinates are then used as weights for the blending between animations. A nice property of the barycentric coordinates is that the sum equals 1, which is a requirement for our blending. Finally in order to move the character towards the next position, we need to displace the root of the character adequately to avoid foot sliding. The final root displacement

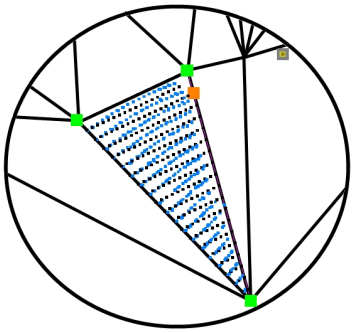


Figure 5: Offsets for different landing positions in a triangle, between barycentric coordinates interpolation (black dots) and blending the whole skeleton using SLERP (blue dots).

vector, $\mathbf{d}_j^r$ is calculated as the weighed sum of the root's displacement of the three selected animation steps (Eq. 2), and changes in orientation of the input root trajectory are applied as rotations over the ball of the supporting foot.

$$\mathbf{d}_j^r = \lambda_1 \cdot \mathbf{d}_1^r + \lambda_2 \cdot \mathbf{d}_2^r + \lambda_3 \cdot \mathbf{d}_3^r \qquad (2)$$

This provides a final root displacement that is the result of interpolating between the three root displacements in order to avoid any foot sliding. It is important to notice that the barycentric coordinates provide the linear interpolation required between three points in 2D space to obtain the position q_j. This is an approximation of the real landing position that our character will reach, as the result of blending the different poses of the three animation clips, using spherical linear interpolation (SLERP) with a simple iterative approach as described in [30].

Therefore there will be an offset between the desired position q_j and the position reached after interpolating the three animations. To illustrate this offset, Fig. 5 shows the points sampled to compute barycentric coordinates in black, and in blue the real landing positions achieved after applying the barycentric weights to the animation engine and performing blending using SLERP. In order to correct this small offset at the same time that we adjust the feet to the elevation of the terrain and orient the footstep correctly, we incorporate a fast and simple IK solver.

4.5. Inverse Kinematics

An analytical IK solver modifies the leg joints in order to reach the desired position at the right time with a pose as close as possible to the original motion capture data. For footstep-based control, the desired foot position is already encoded in the footstep trajectory, and for COM trajectories the final position is calculated by projecting the current position of the foot over the terrain. The controller feeds the IK system with the end position and orientation for each footstep. This allows the system to handle footsteps on uneven terrain.

5. Incorporating Root Movement Fidelity

In some scenarios the user may be more interested in following root velocities than in placing the feet at exact footsteps

or with specific walking styles. We present a solution to include root movement based interpolation in our current barycentric coordinates based interpolator through a user controlled parameter λ_4.

For this purpose, we incorporate the locomotion system presented by Johansen [9] to produce synthesized motions that follow a COM trajectory with correction for uneven terrain. During offline analysis, a parametric space is defined using all the root velocity vectors extracted from the clips in the motion database. For example, a walk forward clip at 1.5 m/s, and a left step clip at 0.5 m/s produces a parametric space using the root velocity vectors going from the forward direction to the 90^o direction, and with speeds from 0.5 m/s to 1.5 m/s.

Given a desired root velocity we define a parametric space Ω_r, and a gradient band interpolator in polar space [9] is created to compute the weights for each animation clip to produce the final blended result. The gradient band interpolator does not ensure accuracy of the produced parameter values but it does ensure smooth interpolation under dynamically and continuously changing parameter values, as with a player-controlled character. Once the different clips are blended with the computed weights, the system predicts the support foot position at the end of the cycle and projects it on the ground to find the exact position where it should land.

The root movement based interpolator will select a set of k animations A_1^r to A_k^r with their corresponding weights: $w_1, ..., w_k$. Each of those animations provides a landing position $p_1^r, ..., p_k^r$, and if we only interpolated these animations we would obtain the landing point r.

In order to incorporate the output of the polar gradient band interpolator in the barycentric coordinates based interpolator we proceed as indicated in Algorithm 1.

The algorithm first checks whether a vertex of the current triangle $\langle p_1, p_2, p_3 \rangle$ can be replaced by any of the three vertices with highest weights selected by the polar band interpolator, p_j^r, $j \in [1, k]$ (lines 1-13 in the algorithm). This replacement takes place if the distance between the two landing positions p_i and p_j^r is within a user input threshold ϵ (line 7), and the resulting triangle still contains the desired landing position q_j (function *IsInTriangle* returns true if q_j is inside the new triangle). This means that there is another animation that also provides a valid triangle and has a root velocity that is closer to the input root velocity.

Next, function *CalculateRootLanding* computes the landing position reached after blending the animations given by the root movement interpolator (Eq. 3).

$$r = \sum_{i=1}^{k} w_i \cdot p_i^r \tag{3}$$

Finally, *ComputeWeights* calculates the three λ_i for the next footstep q_j by incorporating a user provided λ_4 and the result of the polar band interpolator r (Eq. 4).

$$q_j = \lambda_1 \cdot p_1 + \lambda_2 \cdot p_2 + \lambda_3 \cdot p_3 + \lambda_4 \cdot r \tag{4}$$

Algorithm 1 Incorporating root movement fidelity

Input:
- The target position q_j,
- The current triangle $\langle p_1, p_2, p_3 \rangle$,
- Root landing positions $\left\langle p_1^r, ..., p_k^r \right\rangle$,
- Animation weights $\langle w_1, ..., w_k \rangle | w_1 \geq ... \geq w_k$,
- A user input threshold ϵ,
- A user input weight parameter λ_4

Output: $\lambda_1, \lambda_2, \lambda_3$

1: **for** $i = 1$ **to** 3 **do**
2: $u \leftarrow (i + 1) \mod 3$
3: $v \leftarrow (i + 2) \mod 3$
4: $j \leftarrow 1$
5: $replaced \leftarrow$ false
6: **while** $j \leq 3 \wedge \neg replaced$ **do**
7: **if** $\left\| p_i - p_j^r \right\| \leq \epsilon \wedge IsInTriangle\left(q_j, \left\langle p_j^r, p_u, p_v \right\rangle\right)$ **then**
8: $p_i \leftarrow p_j^r$
9: $replaced \leftarrow$ true
10: **end if**
11: $j \leftarrow j + 1$
12: **end while**
13: **end for**
14: $r \leftarrow CalculateRootLanding\left(\left\langle p_1^r, ..., p_k^r \right\rangle, \langle w_1, ..., w_k \rangle\right)$
15: $\langle \lambda_1, \lambda_2, \lambda_3 \rangle \leftarrow ComputeWeights\left(\langle p_1, p_2, p_3 \rangle, \lambda_4, r\right)$

and λ_i are defined using the following relationship:

$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1 \tag{5}$$

Since w_i and p_i^r are known $\forall i \in \{1, ..., k\}$, and λ_4 is a user input, we have a linear system, where λ_4 determines the trade-off between following footsteps accurately (if $\lambda_4 = 0$), and simply following root movement (if $\lambda_4 = 1$).

As the user increases λ_4 there will be a value $\beta \in [0, 1]$ for which λ_1, λ_2 or λ_3 will be negative, when solving the system of equations formed by eq.4 and eq.5. In order to avoid animation artifacts it is necessary to deal only with positive weights, therefore we guarantee that the system will only reproduce q_j accurately as long as $\lambda_4 < \beta$. If we further increase λ_4 beyond the value β then the algorithm will provide the blending values that correspond to a new point q' which is the result of a linear interpolation between q_j and point r. When $\lambda_4 = 1$ the resulting blending will be exclusively the one provided by the root movement trajectory since $\lambda_1 = \lambda_2 = \lambda_3 = 0$. Fig. 6 illustrates this situation.

Time Warping. Incorporating root velocity in the interpolation, does not always guarantee that the time constraints assigned per footstep will be satisfied. Therefore once we have the final set of animations to interpolate between, with their corresponding weights λ_i, $i \in \{1, 2, 3\}$ and w_j, $j \in [1, k]$, we need to apply time warping. Each input footstep f_m has a time stamp τ_m indicating the time at which position q_m should be reached (where $m \in [1, M]$ and M is the number of footsteps in the input trajectory). The total time of the current motion, T can

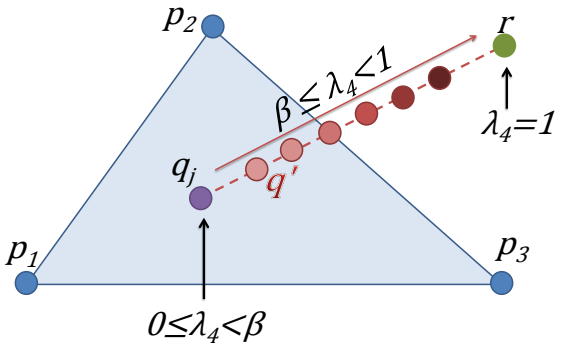


Figure 6: When solving the system of equations given by eq.4 and eq.5, the value of either λ_1, λ_2 or λ_3 will be negative when $\lambda_4 \geq \beta$. Therefore we need to calculate the barycentric coordinates for a new point q'which moves linearly from q_j to r as the user increases the value of λ_4 from β to 1. This means solving the system of equations for q' instead of q_j, as it is the closest point to the desired landing position which guarantees that all weights in eq. 5 will be positive.

be calculated as the weighted sum of the time of the animation steps being interpolated: $T = \sum_{i=1}^{3}(\lambda_i \cdot t(A_i)) + \sum_{j=1}^{k}(w_j \cdot t(A_j))$. Therefore the time warping factor that needs to be applied can be calculated as: $warp_m = (\tau_m - \tau_{m-1})/T$.

Outside the Convex-Hull. The footstep parametric space defines a convex-hull delimiting the area where our character can land its feet. When our target footstep position falls inside this area, clips can be interpolated to reach that desired position. But if it falls outside this convex-hull we still want the system to consider and try to reach it. Our solution to handle this problem consists of projecting orthogonally the input landing position q over the convex-hull to a new position q_{proj}. Our system then gives the blending weights for q_{proj} and applies IK to adjust the final position. We include a parameter to define a maximum distance for the IK to set an upper limit on the correction of the landing position. It is important to notice that even if the input trajectory has some footsteps that are unreachable with the current data base of animation clips, our system will provide a synthesized animation that will follow the input trajectory as closely as possible, until it recovers and catches up with future steps in the input trajectory. This situation is similar to the scenarios where the user increases λ_4 and then reduces it again.

6. Results

The animation system described in the paper is implemented in C# using the Unity 3D Engine [31]. The footstep trajectories used to animate the characters are generated using the method described in [7] or are created by the user. Some difficult scenarios, exercising careful footstep selection, are shown in Fig. 1 and Fig. 7. Agents carefully plant their feet over pillars (Fig. 7-a) or use stepping stones to avoid falling into the water (Fig. 7-b). We show our ability to handle over a hundred agents at 13 FPS (Fig. 7-c and Fig. 9). The supplementary video demonstrates additional results (high resolution video[2], low resolution video[3]).

Obstacle Course. We exercise the locomotion dexterity of a single animated character in an obstacle course. The character follows a footstep trajectory with different walking gaits , alternating running and walking phases (Fig. 1-a,b), and including sidesteps (Fig. 1-c) and backward motion (Fig. 1-e).

Stepping Stone Problem. Stepping stone problems (Fig. 7-b) require careful footstep level precision where constraints require the character to place their feet exactly on top of the stones in order to successfully navigate the environment. Our framework can be coupled with footstep-based controllers to solve these challenging benchmarks.

Integration with Crowd Simulator. We integrate our animation system with footstep-based simulators [7]; our character follows the simulated trajectories without compromising its motion fidelity while scaling to handle large crowds of characters (Fig. 7-c).

It is important to mention that the quality of the results depends strongly on the quality of the clips available from the motion capture library. As can be seen in the video, the least precise movements in our results are side steps and back steps. This is due to two reasons: (1) we had a small number of animations compared to other walking gaits, and thus triangles covering that space have larger areas, and (2) interpolation artifacts appear when blending between animations that move in opposite directions (for example a backwards step with a forward step). We believe that having a better and denser sampling in these areas will improve the results. For steps falling in triangles of smaller areas, and with all the vertices in the same quartile we have obtained results of high quality even for difficult animations such as running or performing small jumps.

6.1. Foot Placement Accuracy

The presented barycentric coordinates interpolator assumes a small offset between the results of linearly interpolating landing positions from the set of animations being blended, and the actual landing position when calculating spherical linear interpolation over the set of quaternions. This small offset depends on the area of the triangle, so as we incorporate more animations into our data base, we obtain a denser sampling of landing positions and thus reduce both the area of the triangles and the offset. We believe this is a convenient trade off since such a small offset can be eliminated with a simple analytical solver but the efficiency of computing barycentric coordinates offers great performance. It is also important to notice that if exact foot location is not necessary, and the user only needs to indicate small areas for stepping as in the water scenario, then it is not necessary to apply the IK correction. Fig. 8 shows the offset between the landing position and the footstep. The magnitude of the error is illustrated as the height of the red cylinders that are located at the exact location where the foot first strikes.

[2]https://www.dropbox.com/s/o1b9w73qd45fmip/videoCAG.mp4?dl=0
[3]https://www.dropbox.com/s/ptdz788f2k9ad3g/videoCAGlowRes.mp4?dl=0

(a) (b) (c)

Figure 7: (a) Agents accurately following a footstep trajectory and avoiding falls by carefully stepping over pillars. (b) The stepping stone problem is solved with characters avoiding falls into the water. (c) A crowd of over 100 agents simulated at interactive rates.

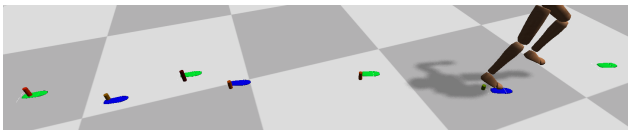


Figure 8: The red columns show the small offset between landing position and the footstep when the IK corrections are not being applied.

6.2. *Performance*

Fig. 9 shows the frame rate we obtain as we double the number of agents. It is important to notice that increasing the number of animations would enhance the quality and accuracy of the results, with just a small overhead on the performance.

The average time of the locomotion controller is 0.43ms, this process includes blending animations, IK, the polar band interpolator and our barycentric coordinates based interpolator. The computational cost of our footstep interpolator is 0.2 ms, which is amortized over several frames as the interpolation in Ω_{f_L} or Ω_{f_R} only need to be performed once per footstep. This time is divided between computing the root movement polar band interpolator which takes 0.155ms and our barycentric coordinates interpolator which takes 0.045ms. Performance results were measured on an Intel Core i7-2600k CPU at 3.40GHz with 16GB RAM.

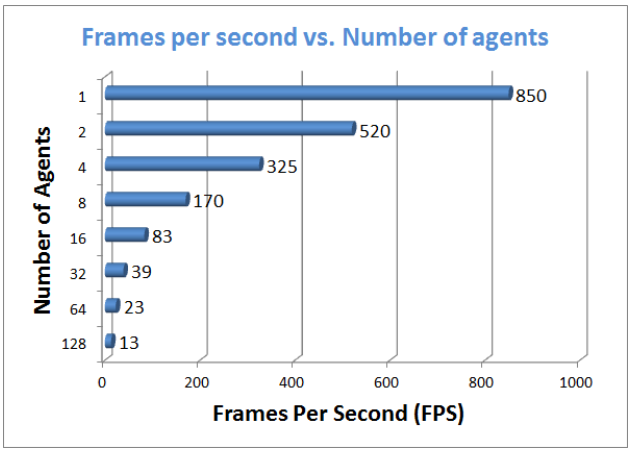


Figure 9: Performance of the Footstep Locomotion System in frames per second as the number of agents increases.

7. Conclusions and Future Work

We have presented a system that uses multiple parameter spaces to animate fully embodied virtual humans to accurately follow a footstep trajectory respecting root velocities, using a relatively small number of animation clips (24 in our examples). Our method is fast enough to be used with tens of characters in real time (25 FPS) and over a hundred characters at 13 FPS. The method can handle uneven terrain, and can be easily extended to introduce additional locomotion behaviors by grouping new sets of animation clips and generating different parametric spaces. For example, walking and running motions can be blended together, but if we wanted to add crawling motions or jumping motions, it would be better to separate them in different parametric spaces for each style. This will avoid unnatural interpolations that can appear when blending between very different styles. Having different parametric spaces requires some sort of classification, which could initially be done manually but it could also be based on the characteristics of the motion, such as changes in acceleration, maximum heights of the root, length of fly phase, etc. Assuming we can extract the parametric spaces for different animation types, it would also be necessary in some cases to have additional transition clips to switch between very different locomotion types, i.e. crawling and walking.

We do not run physical or biomechanical simulations, and use interpolation and blending between motion capture animations. Our method accuracy depends on the variety of animation clips, while its quality and efficiency depends on the number of clips. A trade-off between efficiency and accuracy is therefore necessary, for which we have found a good equilibrium.

Limitations. In order to reduce the dimensionality of the problem, we have not included in our parametric space the orientation of the previous footstep. Ignoring the final orientation of the character at the end of the previous step can induce some discontinuities between footsteps. We mitigate this effect by blending between footsteps automatically for a small amount of time (about 0.2 seconds) at the advantage of reducing the computational time and thus making our method suitable for large groups of agents in real time. Regarding the selection of animation at the end of each footstep, notice that in our database, left and right animation steps are extracted from complete animation cycles that are usually consistent in parameters such as

velocity, acceleration and walking gait. Therefore for a given sequence of steps, the most likely animation steps to be chosen will be those extracted from the same set of animation cycles, thus resulting in smooth and natural transitions between very similar steps. When the characteristics of the steps change drastically, then our method needs to blend between steps from very different animation cycles. So in general, alternating left/right steps results in natural transitions with smooth continuity when blending animations, and only when the input step trajectory changes drastically between each pair of steps, we may observe transitions between animations that feel unnatural. This can happen if the step trajectory is done manually with artifacts due to the user's lack of experience creating footstep trajectories, or for example when the input trajectory forces the character to walk over artificially located steps, like crossing a river by stepping over stones. We would like to empathize that this situation would also look awkward in the real world and thus the result of our synthesized animation may be the desired one.

Future Work. For future work we would like to extend our barycentric coordinates interpolator to 3D space with the third coordinate being the root velocity. This will free our system from the polar band interpolator which not only takes longer to compute but also selects too many animations which results in slower blending. One thing to explore could be to interleave the execution of the Footstep-based Locomotion Controller from different characters in different frames, ensuring we do not execute it for all the agents in the crowd.

Acknowledgements

This work has been partially funded by the Spanish Ministry of Science and Innovation under Grant TIN2010-20590-C02-01. A. Beacco is also supported by the grant FPUAP2009-2195 (Spanish Ministry of Education). The research reported in this document/presentation was also performed in connection with Contract Number W911NF-10-2-0016 with the U.S. Army Research Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as presenting the official policies or position, either expressed or implied, of the U.S. Army Research Laboratory, or the U.S. Government. Citation of manufacturers or trade names does not constitute an official endorsement or approval of the use thereof. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] C. W. Reynolds, Flocks, herds and schools: A distributed behavioral model, in: ACM SIGGRAPH, Vol. 21, 1987, pp. 25–34.

[2] D. Helbing, I. Farkas, T. Vicsek, Simulating dynamical features of escape panic, Nature 407 (2000) 487–490.

[3] A. Treuille, S. Cooper, Z. Popović, Continuum crowds, in: ACM Transactions On Graphics, Vol. 25, 2006, pp. 1160–1168.

[4] N. Pelechano, J. M. Allbeck, N. I. Badler, Controlling individual agents in high-density crowd simulation, in: ACM Symposium on Computer Animation, 2007, pp. 99–108.

[5] N. Pelechano, B. Spanglang, A. Beacco, Avatar Locomotion in Crowd Simulation, International Journal of Virtual Reality 10 (2011) 13–19.

[6] A. Egges, B. van Basten, One Step at a Time: Animating Virtual Characters Based on Foot Placement, The Visual Computer 26 (6-8) (2010) 497–503.

[7] S. Singh, M. Kapadia, G. Reinman, P. Faloutsos, Footstep Navigation for Dynamic Crowds, Computer Animation and Virtual Worlds 22 (2011) 151–158.

[8] A. Treuille, Y. Lee, Z. Popović, Near-Optimal Character Animation with Continuous Control, ACM Transactions on Graphics 26 (3) (2007) 7.

[9] R. S. Johansen, Automated Semi-Procedural Animation, Master Thesis. URL `http://runevision.com/thesis/`

[10] L. Kovar, M. Gleicher, F. Pighin, Motion graphs, in: ACM SIGGRAPH, 2002, pp. 473–482.

[11] L. Zhao, A. Safonova, Achieving Good Connectivity in Motion Graphs, Graphical Models 71 (4) (2009) 139–152.

[12] C. Ren, L. Zhao, A. Safonova, Human Motion Synthesis with Optimization-Based Graphs, Computer Graphics Forum 29 (2).

[13] J. Min, J. Chai, Motion Graphs++, ACM Transactions On Graphics 31 (6) (2012) 1.

[14] M. Lau, J. J. Kuffner, Precomputed search trees: planning for interactive goal-driven animation, in: ACM Symposium on Computer Animation, 2006, pp. 299–308.

[15] A. Witkin, Z. Popovic, Motion warping, in: ACM SIGGRAPH, 1995, pp. 105–108.

[16] S. Chenney, Flow tiles, in: ACM Symposium on Computer Animation, 2004, pp. 233–242.

[17] A. Sud, E. Andersen, S. Curtis, M. Lin, D. Manocha, Real-time path planning for virtual agents in dynamic environments, in: IEEE Virtual Reality, 2007, pp. 91–98.

[18] P. Glardon, R. Boulic, D. Thalmann, Robust on-line adaptive footplant detection and enforcement for locomotion, The Visual Computer 22 (3) (2006) 194–209.

[19] J. Pettré, J.-P. Laumond, T. Siméon, A 2-stages locomotion planner for digital actors, in: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '03, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2003, pp. 258–264.

[20] J. Pettre, J.-P. Laumond, A motion capture-based control-space approach for walking mannequins: Research articles, Computer Animation and Virtual Worlds 17 (2) (2006) 109–126.

[21] M. Felis, K. Mombaur, Using Optimal Control Methods to Generate Human Walking Motions, Motion in Games (2012) 197–207.

[22] Autodesk, 3d studio max, www.autodesk.com/products/autodesk-3ds-max/overview (2014).

[23] M. Girard, A. A. Maciejewski, Computational modeling for the computer animation of legged figures, in: SIGGRAPH, ACM, 1985, pp. 263–270.

[24] J. Chai, J. K. Hodgins, Constraint-Based Motion Optimization Using A Statistical Dynamic Model, ACM SIGGRAPH.

[25] M. G. Choi, J. Lee, S. Y. Shin, Planning biped locomotion using motion capture data and probabilistic roadmaps, ACM Transactions on Graphics 22 (2) (2003) 182–203.

[26] H. Ko, N. I. Badler, Animating human locomotion with inverse dynamics, IEEE Computer Graphics & Applications 16 (2) (1996) 50–59.

[27] M. van de Panne, From Footprints to Animation, Computer Graphics Forum 16 (4) (1997) 211–223.

[28] B. van Basten, P. W. A. M. Peeters, A. Egges, The Step Space : Example-Based Footprint-Driven Motion Synthesis, Computer Animation and Virtual Worlds 21 (May) (2010) 433–441.

[29] B. van Basten, S. Stüvel, A. Egges, A Hybrid Interpolation Scheme for Footprint-Driven Walking Synthesis, Graphics Interface (2011) 9–16.

[30] A. Shoulson, N. Marshak, M. Kapadia, N. I. Badler, ADAPT : The Agent Development and Prototyping Testbed, ACM SIGGRAPH I3D.

[31] Unity, Unity - game engine (2014). URL `http://unity3d.com/`