WarpDriver: Context-Aware Probabilistic Motion Prediction for Crowd Simulation

David Wolinski[*] INRIA
Ming C. Lin[†] UNC-Chapel Hill
Julien Pettré[‡] INRIA

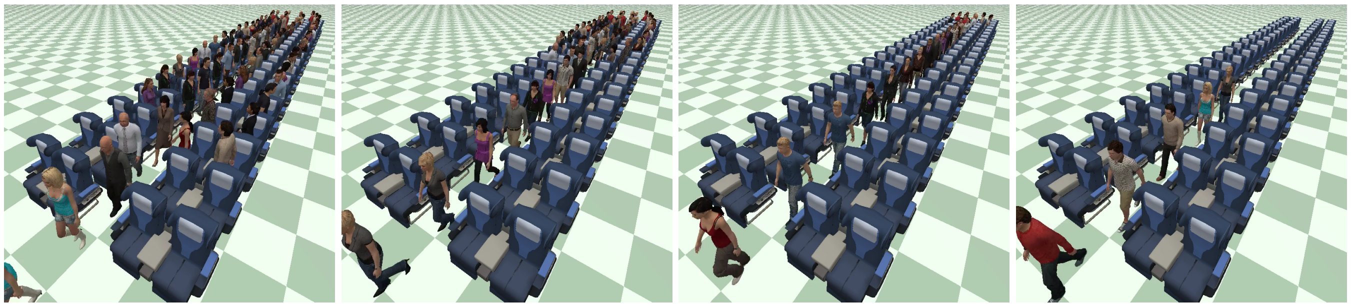


Figure 1: *WarpDriver agents exiting a plane in seating order, solely due to collision avoidance, without additional scripting.*

Abstract

Microscopic crowd simulators rely on models of local interaction (e.g. collision avoidance) to synthesize the individual motion of each virtual agent. The quality of the resulting motions heavily depends on this component, which has significantly improved in the past few years. Recent advances have been in particular due to the introduction of a short-horizon motion prediction strategy that enables anticipated motion adaptation during local interactions among agents. However, the simplicity of prediction techniques of existing models somewhat limits their domain of validity. In this paper, our key objective is to significantly improve the quality of simulations by expanding the applicable range of motion predictions. To this end, we present a novel local interaction algorithm with a new context-aware, probabilistic motion prediction model. By context-aware, we mean that this approach allows crowd simulators to account for many factors, such as the influence of environment layouts or in-progress interactions among agents, and has the ability to simultaneously maintain several possible alternate scenarios for future motions and to cope with uncertainties on sensing and other agent's motions. Technically, this model introduces "collision probability fields" between agents, efficiently computed through the cumulative application of *Warp Operators* on a source *Intrinsic Field*. We demonstrate how this model significantly improves the quality of simulated motions in challenging scenarios, such as dense crowds and complex environments.

Keywords: crowd simulation, anticipation, collision avoidance

Concepts: •**Computing methodologies** → **Agent / discrete models;** Procedural animation;

[*]e-mail:david.wolinski@inria.fr
[†]e-mail:lin@cs.unc.edu
[‡]e-mail:julien.pettre@inria.fr

SA '16 Technical Papers,, December 05-08, 2016, Macao
ISBN: 978-1-4503-4514-9/16/12
DOI: http://dx.doi.org/10.1145/2980179.2982442

1 Introduction

Much attention has recently been devoted to crowd simulation due to its applications in pedestrian dynamics, virtual reality and digital entertainment. As a result, many algorithms have been proposed and they are typically separated into two main classes: macroscopic algorithms that simulate crowds as a whole, and microscopic algorithms that model individual movement. Algorithms of this second type can generate realistic *individual* agent trajectories and this capability is important for most crowd applications. At their core, microscopic crowd simulators rely on the notion of a local interaction model to formulate how agents influence each other's trajectory. The most required model of local interactions deals with collision avoidance between agents which is the focus of our paper. The quality of resulting simulations directly depends on these models because, when numerous interactions occur such as in crowds, they mostly determine how individual trajectories are formed. As detailed in the next section, most recent approaches rely on a short-term motion prediction mechanism in order to anticipate motion adaptations during local interactions. They are referred to as velocity-based algorithms as this prediction relies on the current positions and velocities of agents. This new principle for interaction models allowed for significant progress in terms of realism at both the local and global levels, because anticipation is observed in humans. Despite these important advances, some issues persist and have direct impact on simulation results.

Our hypothesis is that the persisting issues are due to a few basic assumptions in the design of these local interaction models. In particular, existing algorithms often assume that the current velocity of agents is representative of their motion intent, and their motion prediction relies on the assumption of a constant velocity. Obviously, the current velocity of agents could not always be representative of their intent, for instance, when an agent turns or adapts its motion to avoid collisions. Section 7 shows scenarios where prediction based on simple linear motion extrapolation fails. Capturing a wider set of observations on how each agent determines its motion, it is possible to make more accurate motion predictions and consequently to simulate more realistic local agent interactions. This realization is the key insight in this paper. We propose a stochastic motion prediction model that accounts for the "context" of local agent-agent and agent-environment interactions.

More precisely, two main aspects distinguish our solution from previous ones. The first is our representation of future events. In previous work, this is based on a simple linear extrapolation of each agent's current velocity. In our model, each agent constructs a (possibly non-linear) probability field of colliding with other agents. This representation is versatile, as it allows the crowd simulation to maintain multiple possible future motions (more or less probable) or to model various uncertainties due to sensing and human behaviors in motion predictions. The second aspect is the model of how agents react to this motion prediction. Given the probability fields of collisions, the local collision response and avoidance can be computed using a gradient descent on these fields. This solution differs from macroscopic algorithms (e.g. [Treuille et al. 2006]), whose density fields do not model agents' future motions, sensing uncertainties, or other agents' responses.

In this paper, we introduce a generalized space-time local interaction model for crowd simulation using a unified, probabilistic theoretic framework accounting for stochastic (and possibly nonlinear) motion prediction, non-deterministic sensing, and the unpredictability of human behaviors. Our main contributions include:

- A new collision avoidance algorithm that relies on a probabilistic prediction of each agent's future motions.
- A technique to efficiently compute future collision probabilities thanks to an *Intrinsic Field* and *Warp Operators*; consequently, we refer to this algorithm as "WarpDriver".

The rest of the paper is organized as follows. Section 2 provides a brief review of related work and Sections 3, 4, 5, and 6 are devoted to the technical description of our approach. In Section 7, we demonstrate the benefits of our algorithm as compared to some of the most recent algorithms, and discuss existing artifacts in highly dense crowds, circulation in dynamic and complex environments, and interactions with erratically-behaved agents. We show how this new context-aware probabilistic motion prediction model can alleviate many of these commonly known issues of existing simulation.

2 Related Work

Much attention has recently been devoted to crowd simulation due to its applications in pedestrian dynamics, virtual reality and cinematic entertainment. Consequently, many crowd simulation algorithms, spanning several categories and each with its own characteristics, have been devised. Macroscopic crowd simulation algorithms [Narain et al. 2009; Treuille et al. 2006] animate crowds at the global level, aiming to capture statistical quantities such as flows or densities. In contrast, microscopic algorithms model interactions between individual pedestrians, with the emergence of movement patterns at the crowd level. For instance, algorithms based on cellular automata discretize space into grids where pedestrians are moved based on transition probabilities [Kretz and Schreckenberg 2008; Schadschneider 2001]. Other, agent-based algorithms model pedestrians as agents, with various levels of complexity. Finally, example-based algorithms maintain databases of crowd motions which can be reused depending on the context [Lerner et al. 2007; Ju et al. 2010].

Among these works, agent-based algorithms remain very popular, due to their ease of implementation and their flexibility through various extensions and scripting. To reproduce local interactions between people, these algorithms have always focused on the most readily available and easily useable information: agents' positions. This has been the case starting with Reynolds' seminal work with the Boids algorithm [Reynolds 1987], later the Social-Forces algorithm by [Helbing and Molnár 1995; Helbing et al. 2000] and many of their derivatives ever since.

However, anticipation of each other's trajectories is key to people's interactions, and efficient, collision-free navigation [Olivier et al. 2012; Karamouzas et al. 2014]. In light of this observation, major advances recently came from velocity-based algorithms. [Reynolds 1999] introduced the point of closest approach between agents, where, if the distance between the concerned agents was to be low enough at this point (reflecting a collision), they would steer away from it. Later, in an algorithm derived from Social-Forces, [Karamouzas et al. 2009] used this point of closest approach as a source of repulsive forces; and [Pellegrini et al. 2009] used the distance of the closest approach to refrain from choosing velocities which might lead to collisions. In parallel, other algorithms [Feurtey 2000; Paris et al. 2007] work in space-time (2-dimensional space plus one more dimension of time) to, again, select permitted, collision-free velocities. This method of choosing permissible velocities was further accelerated by algorithms that reasoned in 2-dimensional velocity-space such as [van den Berg et al. 2008; Guy et al. 2009; Guy et al. 2012a; Pettré et al. 2009]. Most recently, [Karamouzas et al. 2014] introduced an algorithm where velocity-based interactions are formulated as an optimization problem, the parameters of which are derived from observation data, similarly to [Liu et al. 2005]. Finally, other algorithms used instantaneous velocities in other ways, such as affordance fields [Kapadia et al. 2009] and velocity-derived values processed from the synthetic visual flows of agents [Ondřej et al. 2010].

As a common assumption, these algorithms all linearly extrapolate agents' future motions from their positions and velocities, making it possible to anticipate collisions up to a certain time horizon and improve simulation results [Olivier et al. 2012; Guy et al. 2012b; Wolinski et al. 2014]. However, this linear extrapolation remains simplistic, and in many more challenging situations, does not yield truly satisfactory results. Consequently, [Kim et al. 2014] introduced a probabilistic component to the algorithm presented by [van den Berg et al. 2008], while [Golas et al. 2013] added look-ahead to adaptively increase the time horizon in an efficient way for large groups. Finally, [van den Berg et al. 2011a] incorporated agents' acceleration constraints into this same algorithm.

This underlying assumption of linear motion prediction, however, does not hold in many cases, and we suggest that constraining a crowd simulator to only information on positions and instantaneous velocities is often insufficient. By addressing these issues, we introduce an approach that enables agents to efficiently take into account arbitrary sources of information in a stochastic framework when anticipating each other's future motions.

3 Overview

Our algorithm builds on an agent-based modeling framework and the resulting simulator captures complex interactions among agents. In this section, we provide a high-level overview (Figure 2) of our approach, i.e. how we model interactions between agents and steer them. However, before describing "WarpDriver", we first need to define what we consider an agent in our formulation. An agent is any entity that the algorithm would steer or any other entity that could affect another agent's steering decisions. Agents can be, for instance, pedestrians, cars or walls, and they can further have various *properties*: size, shape, position, velocity, followed path, etc. In addition, in our formulation, interactions between agents are resolved in space-time. To simulate these interactions, we identify the *perceiving* agent (the agent we are currently steering) and the *perceived* agents (the agents that are to be avoided). Interactions among agents are modeled in three main steps:

Step 1, Setup: The *perceiving* agent starts by defining its space-time *projected trajectory*: the trajectory it would follow if no collisions were to happen (red dotted line on left of Figure 2 and Figure 3; detailed in Section 4).

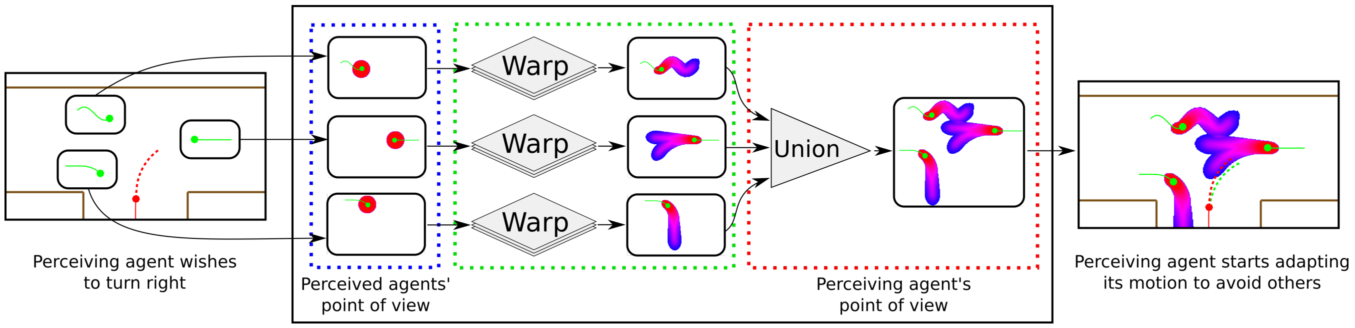

Warp
Warp
Warp
Union
Perceiving agent wishes to turn right
Perceived agents' point of view
Perceiving agent's point of view
Perceiving agent starts adapting its motion to avoid others

Figure 2: *Overview of the algorithmic framework of WarpDriver.*

Step 2, Perceive: This agent then constructs its perception of other *perceived* agents' future motions in the form of space-time collision probabilities (middle of Figure 2 and color gradient on Figure 3; detailed in Section 5).

Step 3, Solve: Finally, the agent intersects its *projected trajectory* with these probabilities (thus evaluating the chances of collision along the *projected trajectory*) and modifies its *projected trajectory* by performing one step of gradient descent to lower its collision probabilities along this trajectory (green dotted line on right of Figure 2 and Figure 3; detailed in Section 6).

The most important aspect of our approach is then how the *perceiving* agent derives collision probabilities from the *perceived* agents. It is through this process that we can model any non-linear behavior of both *perceiving* and *perceived* agents.

Our goal for the collision probability formulation process (Step 2) is the ability to handle each *property* separately. Thus, we define the *Intrinsic Field* as the lowest common denominator among agents: the fact that they occupy a volume in space-time (they co-exist); this is a collision probability field. We then model any additional *property* as a *Warp Operator* which further warps the *Intrinsic Field*.

In order to define a clean system pipeline for implementation, we further associate every agent with its own *agent-centric* space-time. Step 2 is then described by the following three sub-steps:

- Every *perceived* agent is modeled as an *Intrinsic Field* in its *agent-centric* space-time (blue rectangle in Figure 2).
- *Warp Operators* progressively warp every *perceived* agent's *Intrinsic Field* from its *agent-centric* space-time into the *perceiving* agent's *agent-centric* space-time (green rectangle in Figure 2).
- These warped collision probability fields (in the *perceiving* agent's *agent-centric* space-time) are then combined into a single collision probability field (red rectangle in Figure 2).

Note that by confining agents' *properties* to Step 2, the *perceiving* agent's *projected trajectory* can simply be a line in its *agent-centric* space-time, simplifying further computations (Sections 4, 5, 6).

4 Notations and Setup

In this section, we describe the notations used throughout the paper and detail how a *perceiving* agent constructs its *projected trajectory*, i.e. its current trajectory in space-time assuming no collisions take place (Step 1 of our approach, see Figure 2):

- $\cdot, \times$, $\circ$, $\star$ and $*$ respectively denote the dot product, cross product, function composition, component-wise multiplication and convolution.

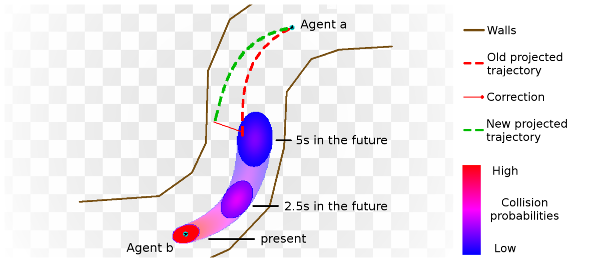


Figure 3: *Illustration of collision avoidance between two agents a and b on a curved path. The color gradient represents a's probability of collision with b, as perceived by a. The red dotted line represents a's initial* projected trajectory. *The green dotted line represents a's final, corrected,* projected trajectory *(exaggerated). The red line in between both dotted ones represents the correction agent a will perform (exaggerated for illustration). Note that the* projected trajectory *is a curve path due to* Warp Operators.

- $\mathcal{A}$ is the set of all agents, $a, b \in \mathcal{A}$ are two such agents; note that a usually denotes the *perceiving* agent while b usually denotes the *perceived* agent.
- $\mathcal{S}$ is a 3D space-time with basis $\{\mathbf{x}, \mathbf{y}, \mathbf{t}\}$, where $\mathbf{x}$ and $\mathbf{y}$ form 2D positions and $\mathbf{t}$ is the time. $\forall s \in \mathcal{S}, \exists\ x, y, t \in \mathbb{R}/s = x.\mathbf{x} + y.\mathbf{y} + t.\mathbf{t}$ Note the difference between bold-face vectors (e.g. $\mathbf{x}$) and normal-font scalar quantities (e.g. x).
- $\mathcal{S}_{a,k}$ is the *agent-centric* space-time $\mathcal{S}$ centered on an agent a at timestep k such that, in this space-time, agent a is at position $\mathbf{o} = (0,\ 0,\ 0) \in \mathcal{S}_{a,k}$ and faces along the local $\mathbf{x}$ axis, positive values along the local $\mathbf{t}$ axis represent the future.
- ∇ is the nabla operator. For a continuous scalar field f over an $\mathcal{S}$-like space, ∇f is the gradient of f.
- $\mathbf{r}_{a,k}$ is agent a's *projected trajectory* in $\mathcal{S}_{a,k}$.
- $\forall \mathbf{s} \in \mathcal{S}_{a,k},\ p_{a \to b,k}(\mathbf{s})$ is what agent a perceives to be its collision probability with agent b.
- I, the *Intrinsic Field*, gives the probability of colliding with any agent b in space-time $\mathcal{S}_{b,k}$. ∇I is the gradient of I.
- W denotes a *Warp Operator* that warps I for every *property* of an agent. $\mathbf{W} = W_n \circ ... \circ W_1$ further denotes the composition of operators $\{W_1, ..., W_n\}$.
- W^{-1} is used to apply the inverse of a *Warp Operator* W to probabilities and probability gradients. Assuming a collection of operators $\{W_1, ..., W_n\}$ where $\mathbf{W}(\mathcal{S}_{a,k}) = \mathcal{S}_{b,k}$, then $\mathbf{W}^{-1} = W_1^{-1} \circ ... \circ W_n^{-1}$ and $\forall \mathbf{s} \in \mathcal{S}_{a,k}$:

$$(\mathbf{W}^{-1} \circ I \circ \mathbf{W})(\mathbf{s}) = p_{a \to b,k}(\mathbf{s}), \tag{1}$$

$$(\mathbf{W}^{-1} \circ (\nabla I) \circ \mathbf{W})(\mathbf{s}) = \nabla p_{a \to b,k}(\mathbf{s}). \tag{2}$$

With these notations, in Step 1 of our approach, the *perceiving* agent a constructs its *projected trajectory* $\mathbf{r}_{a,k}$ in its *agent-centric* space-time $\mathcal{S}_{a,k}$. We further assume that the *perceiving* agent a is a point in its *agent-centric* space-time, $\mathcal{S}_{a,k}$ is then its configuration-space. As mentioned in Section 3, since the processing of agents' *properties* is confined to Step 2, the *perceiving* agent's *projected trajectory* can be defined as a line.

Specifically, assuming agent a has an instantaneous speed $v_{a,k}$ at timestep k, its *projected trajectory* is $\mathbf{r}_{a,k} = line(\mathbf{o},\ v_{a,k}\mathbf{x} + \mathbf{t})$ (a line passing through the origin $\mathbf{o}$ and directed by vector $v_{a,k}\mathbf{x} + \mathbf{t}$). Thus, at any time $t \in \mathbb{R}$ in the future, the *perceiving* agent a projects to be at point $\mathbf{r}_{a,k}(t) = \mathbf{o} + t(v_{a,k}\mathbf{x} + \mathbf{t})$ in space-time $\mathcal{S}_{a,k}$.

5 Perception: collision probability Fields

We here describe how the *perceiving* agent constructs collision probabilities from the *perceived* agents. As mentioned in Section 3, this is a three-step process where:

- the *Intrinsic Field* I is defined for each *perceived* agent b in its *agent-centric* space-time $\mathcal{S}_{b,k}$,
- *Warp Operators* warp I from each $\mathcal{S}_{b,k}$ into the *perceiving* agent a's *agent-centric* space-time $\mathcal{S}_{a,k}$, thus modeling agents' *properties*,
- the resulting collision probability fields are combined.

We detail each of these three steps in the following sub-sections.

5.1 The Intrinsic Field

As defined in Section 3, the *Intrinsic Field* is the lowest common denominator between agents, independently of their *properties*. It is also a continuous collision probability field: for each point $\mathbf{s}$ in a *perceived* agent b's *agent-centric* space-time $\mathcal{S}_{b,k}$, it gives the probability of colliding with b at that point $I(\mathbf{s}) \in [0,\ 1]$.

Since the *perceiving* agent a is a point in its *agent-centric* configuration-space $\mathcal{S}_{a,k}$, any *perceived* agent b should therefore be perceived as a configuration-space obstacle (the Minkowski sum of agents a and b). As we want the *Intrinsic Field* to be independent of agents' *properties* (including size and shape) we define the Minkowski sum of agents a and b as a disk with a normalized radius of 1, this is the step function g:

$$\forall x, y \in \mathbb{R},\ g(x,y) = \begin{cases} 1, & \text{if } \sqrt{x^2+y^2} \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

We further model the perception error with a bivariate normal density function: $\forall x, y \in \mathbb{R},\ f(x,y) = \frac{1}{2\pi} exp(-(\frac{x^2+y^2}{2}))$.

Consequently, we define the *Intrinsic Field* as the convolution of functions f and g:

$$\forall \mathbf{s} \in \mathcal{S}_{b,k},\ I(\mathbf{s}) = (f * g)(\mathbf{s}.\mathbf{x}, \mathbf{s}.\mathbf{y}). \tag{3}$$

It is computed up to a normalized time of 1 second in the future.

5.2 Warp Operators

Warp Operators model each agent *property* that we want to include in the algorithm. As mentioned in Section 3, these could be: shape, size, position, velocity, followed path, etc. Mechanically, *Warp Operators* warp the *Intrinsic Field* defined for each *perceived* agent b in its *agent-centric* space-time $\mathcal{S}_{b,k}$ into the *perceiving* agent a's *agent-centric* space-time $\mathcal{S}_{a,k}$.

In this sub-section, we describe *Warp Operators* modeling agent-related and context-related *properties* (more details in Appendix A).

5.2.1 Agent-Related Operators

The following *Warp Operators* model *properties* which only depend on agents:

Position and Orientation The *Warp Operator* W_{local} models the agents' position and orientation *properties*. It is a simple change of referential between $\mathcal{S}_{a,k}$ and $\mathcal{S}_{b,k}$.

Time Horizon To avoid collisions in a time horizon $\mathcal{T}$ (beyond the normalized 1 second in the *Intrinsic Field*), we define a time horizon operator W_{th}.

Time Uncertainty The W_{tu} operators models the increased uncertainty on the states of other agents the further we look in time.

Radius The W_r operator changes the radius of the agents by dilating space along the $\mathbf{x}$ and $\mathbf{y}$ axes.

Velocity The W_v operator models the agent's instantaneous velocity as a displacement along the $\mathbf{x}$ axis.

Velocity Uncertainty Depending on the speed of an agent, that agent could be more or less likely to make certain adaptations to its trajectory. For instance, the faster an agent travels, the more likely it is to accelerate/decelerate rather that turn. This is modeled by the W_{vu} operator.

5.2.2 Context-Related Operators

The following operators provide information based on the Environment Layout (operator W_{el}), Interactions with Obstacles (operator W_{io}) and Observed Behaviors of agents (operator W_{ob}). These operators, where applicable, **replace** the *Local Space operator* W_{local}. We call W_{ref} the resulting operator: $W_{ref} = \{W_{local}\ or\ W_{el}\ or\ W_{io}\ or\ W_{ob}\ \}$. Note that W_{ref} can be decomposed into a passage from the *perceiving* agent's referential to world referential, and then from world to the *perceived* agent's referential. Any operations in that first part affect the *projected trajectory*.

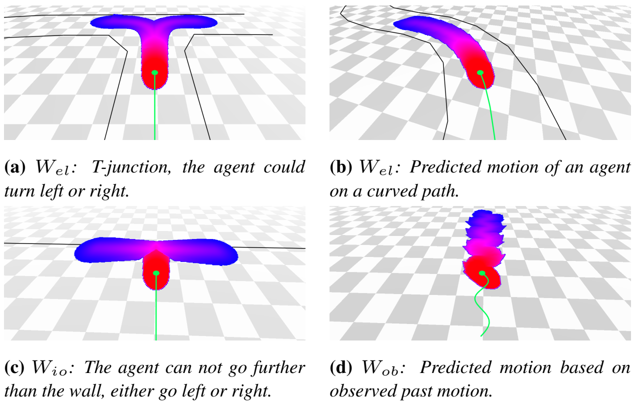

(a) W_{el}: *T-junction, the agent could turn left or right.* **(b)** W_{el}: *Predicted motion of an agent on a curved path.* **(c)** W_{io}: *The agent can not go further than the wall, either go left or right.* **(d)** W_{ob}: *Predicted motion based on observed past motion.*

Figure 4: *Cases using context-related* Warp Operators *(Section 5.2.2). Each case represents one context-related* Warp Operator *combined with all agent-related ones (Section 5.2.1). Same simplified 2D representation as in Figure 3(right).*

Environment Layout When navigating in an environment, based on its layout, we can predict what trajectories other pedestrians are likely to follow. In a series of hallways, for instance, when not threatened by collisions with other pedestrians, one would stay roughly in the middle of the hallway and take smooth turning trajectories at intersections (an agent could turn either left or right in Figure 4a). When navigating on curved paths, one would, again, have a tendency to stay roughly in the middle, resulting in a curved trajectory (Figure 4b). The operator W_{el} models this knowledge by warping space to "align" it with these probable trajectories.

Interactions With Obstacles Where the environment layout operator focuses on agents' probable trajectories assuming they continue travelling on their paths, W_{io} takes care of possible interactions between agents and obstacles. These interactions are essentially much more drastic changes to an agent's locomotion than paths, such as full stops. These can occur if, for instance, an agent comes up to a wall (Figure 4c) (to interact with an ATM, look out the window, check a map...). This can also happen with an agent encountering a small/temporary/unexpected obstacle which could force it to stop and "hug" the obstacle to get around it.

To achieve this, we construct a graph around each obstacle (an obstacle being modeled as a series of connected line segments). When an agent's *projected trajectory* intersects with an obstacle, we extend the graph to that agent and "align" space-time with this graph.

Observed Behaviors With the last operator W_{ob}, we aim to improve the prediction of agents' future motions by looking at their past ones. In the worst case, we might not find any useful information, which won't impact the prediction. However, we might also find some behaviors similar to what the agent is currently doing (e.g. turning in a particular way) or, in the best case, we might find patterns (e.g. agents going in near-circles, zig-zags...) that we can extend to the currently-observed situation (Figure 4d shows anticipation on a zig-zagging agent).

In order to account for this information for an agent a at timestep k, we use a simple method which offers a good tradeoff between cost and results: we keep a history of its positions during h previous timesteps. These past positions form a graph which we repeat on the agent's current position and then "align" space-time with it.

5.2.3 Composition of Warp Operators

As defined in Section 3, we can compose all these operators $\{W_{ref}, W_{th}, W_{tu}, W_r, W_v, W_{vu}\}$:

$$\mathbf{W} = W_{th} \circ W_{tu} \circ W_r \circ W_{vu} \circ W_v \circ W_{ref},$$
$$\mathbf{W}^{-1} = W_{ref}^{-1} \circ W_v^{-1} \circ W_{vu}^{-1} \circ W_r^{-1} \circ W_{tu}^{-1} \circ W_{th}^{-1}.$$

For any point $\mathbf{s}$ in *perceiving* agent a's *agent-centric* space-time $\mathcal{S}_{a,k}$:

$$p_{a\to b,k}(\mathbf{s}) = (\mathbf{W}^{-1} \circ I \circ \mathbf{W})(\mathbf{s}),$$
$$\nabla p_{a\to b,k}(\mathbf{s}) = (\mathbf{W}^{-1} \circ (\nabla I) \circ \mathbf{W})(\mathbf{s}).$$

5.3 Combining collision probability Fields

Before the collision avoidance problem can be solved, one last mechanic still needs to be defined which is how pair-wise interactions can be combined (Step 3 on Figure 2). Let a be the *perceiving* agent, and $b, c \in \mathcal{A},\ b \neq a,\ c \neq a$ be a pair of *perceived* agents. At timestep k, we have access to the following collision probabilities: $p_{a\to b,k}$ and $p_{a\to c,k}$. We can then define the probability agent a has of colliding with either b or c:

$$p_{a\to\{b,c\},k} = p_{a\to b,k} + p_{a\to c,k} - p_{a\to b,k}p_{a\to c,k}.$$

And we can similarly define its gradient:

$$\begin{aligned}\nabla p_{a\to\{b,c\},k} &= \nabla p_{a\to b,k} + \nabla p_{a\to c,k}\\ &\quad - p_{a\to b,k}\nabla p_{a\to c,k}\\ &\quad - p_{a\to c,k}\nabla p_{a\to b,k}\end{aligned}$$

Finally, considering the whole set of agents $\mathcal{A}$, the probability agent a has of colliding with any other agent $b \in \mathcal{A} \setminus a$ is obtained in the same manner, and noted $p_{a\to\mathcal{A}\setminus a,k}$ (with gradient $\nabla p_{a\to\mathcal{A}\setminus a,k}$).

6 Solving the Collision-Avoidance Problem

This section details the third and final step in our approach: how the *perceiving* agent modifies its *projected trajectory* to reduce the collision probabilities along it.

To solve the collision-avoidance problem, the *perceiving* agent samples collision probabilities and their gradients at points $\mathbf{r}_{a,k}(t),\ t \in \mathbb{R}$ along its *projected trajectory* $\mathbf{r}_{a,k}$. First, by averaging these (1) probabilities, (2) gradients, and (3) points, weighted by the probabilities, we respectively compute: (1) the overall collision probability $p_{a,k}$ (the cost function), (2) the associated gradient $\nabla p_{a,k}$, as well as (3) the application point $\mathbf{s}_{a,k}$. We compute these quantities for a time horizon $\mathcal{T}^*$ until a collision with a wall is detected: $\mathcal{T}^* \leq \mathcal{T}$. With the normalization factor $N_{a,k}$, and $t \in [0, \mathcal{T}^*]$:

$$N_{a,k} = \int_t p_{a\to\mathcal{A}\setminus a,k}(\mathbf{r}_{a,k}(t)), \tag{4}$$

we compute $p_{a,k}$, $\nabla p_{a,k}$ and $\mathbf{s}_{a,k}$:

$$p_{a,k} = \frac{1}{N_{a,k}} \int_t p_{a\to\mathcal{A}\setminus a,k}(\mathbf{r}_{a,k}(t))^2, \tag{5}$$

$$\nabla p_{a,k} = \frac{1}{N_{a,k}} \int_t p_{a\to\mathcal{A}\setminus a,k}(\mathbf{r}_{a,k}(t))\ (\nabla p_{a\to\mathcal{A}\setminus a,k})(\mathbf{r}_{a,k}(t)), \tag{6}$$

$$\mathbf{s}_{a,k} = \frac{1}{N_{a,k}} \int_t p_{a\to\mathcal{A}\setminus a,k}(\mathbf{r}_{a,k}(t))\ \mathbf{r}_{a,k}(t). \tag{7}$$

Then, given a user-set parameter α, we move the application point counter to the probability gradient (red line in Figure 3), and use it to define the new trajectory $\mathbf{r}^*_{a,k}$ that agent a should follow in $\mathcal{S}_{a,k}$ to lower its collision probability (green dotted curve in Figure 3):

$$\mathbf{r}^*_{a,k} = line(\mathbf{o},\ \mathbf{s}_{a,k} - \alpha p_{a,k}\nabla p_{a,k}). \tag{8}$$

Additional implementation details can be found in Appendix B.

7 Results

In this section, we show the benefits of WarpDriver as compared with several existing methods. To illustrate the advantages of the more complex *Warp Operators*, we compare WarpDriver with two velocity-based algorithms: the well-known ORCA algorithm [Van Den Berg et al. 2011b] and the recent Powerlaw algorithm [Karamouzas et al. 2014], as they are representative of what can be achieved with velocity-based approaches. We also compare WarpDriver with two position-based algorithms: Boids [Reynolds 1987] and Social-Forces [Helbing and Molnár 1995].

First, we test WarpDriver in challenging scenarios, including large, dense crowds, scenarios with non-linear routes, history-based anticipation cases and a highly-constrained situation. We show the results of our algorithm vs. Powerlaw, ORCA, and Social-Forces (Boids is omitted here, as in these situations it gives largely similar results as Social-Forces). All simulations can be observed on the companion video. Second, we present benchmark results on previously studied data sets for all five algorithms, as well as details on the algorithmic performance of WarpDriver.

Finally, several of the shown values are measured over the duration of the simulated scenarios for each algorithm; in the interest of space, we show these results in a compact way (violin plots, boxplots); the corresponding full graphs can be found in Appendix C.

7.1 Large and Dense Crowds

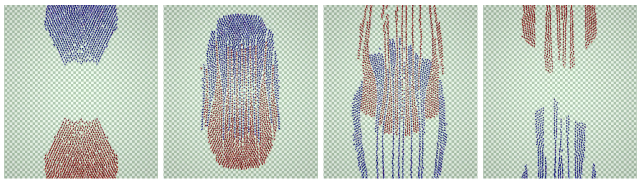


Figure 5: *Big Groups example: two groups of 1027 agents each are made to traverse each other. Simulated with WarpDriver.*

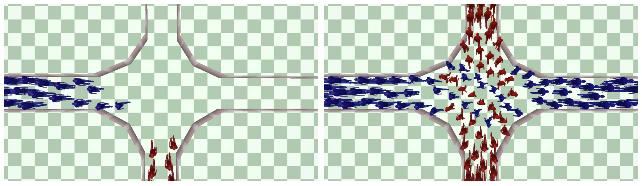


Figure 6: *Crossing example: two flows of agents in corridors cross each other at a right angle (red ones going to the top and blue ones going to the right). Simulated with WarpDriver.*

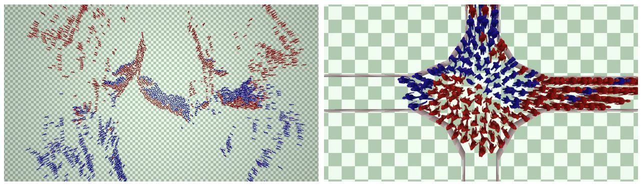


Figure 7: *Issues encountered in Big Groups and Crossing. Left: Big Groups, ORCA agents block each other. Right: Crossing, congestion observed for Powerlaw.*

We start with simulation tests involving a large number of agents and high densities (agents are within contact distance of each other), testing our algorithm's ability to navigate agents while subject to many, simultaneous interactions.

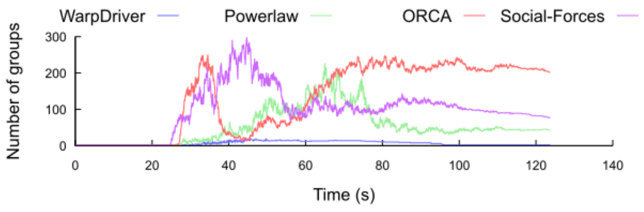
7.1.1 Description

Test case 1: Big Groups This first test case involves two 1027-agent groups exchanging positions as seen on Figure 5. In this kind of example, we expect agents to be able to traverse through the opposing group (ideally with the formation of lanes) and reach their destinations. This expected behavior implies a certain level of organization of the agents; thus we measure how many sub-groups emerge (using the method from [Zhou et al. 2012]) and how widely agents might spread (Figure 8, top and bottom respectively).

Test case 2: Crossing The second test case involves two corridors intersecting at a right angle, each with a uni-directional flow of agents (Figure 6). This kind of situation is well studied and 45° lines should form between agents of each flow at the intersection [Cividini et al. 2013], facilitating their movement. We measure this by detecting sub-groups with the previously mentioned method and perform linear regression on the agents, results are reported on Figure 9(middle, bottom). Furthermore, as the situation is very constrained (agents at contact distance from each other with the presence of walls), we also measure how many agents are jammed (travel at less than 0.1m/s) during the simulations, as shown in Figure 9(top).

7.1.2 Analysis

Big Groups In the Big Groups example (Figure 5), agents simulated with our algorithm are able to do two things. First, front-



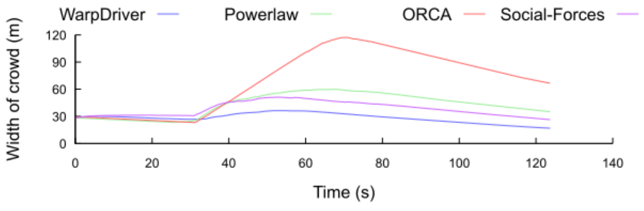
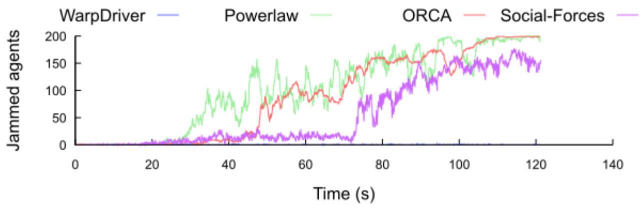


Figure 8: *Top: number of emerging sub-groups, low for WarpDriver (i.e. number of lanes), high for all other algorithms. Bottom: spread of agents, WarpDriver agents stay compact, other agents spread widely.*



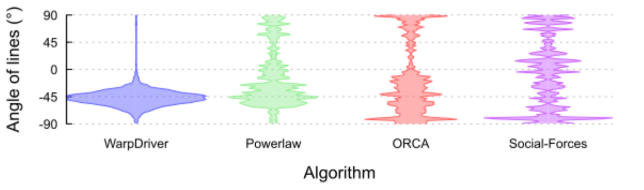


Figure 9: *Top: number of jammed agents, close to none for WarpDriver, many for other algorithms. Bottom: violin plots of detected agent lines at crossing intersection, consistently around −45° for WarpDriver, scattered (and fewer detected lines) for other algorithms; full graphs in Appendix C.*

line agents are able to find points of entry in the opposing group (which correspond to the minima of the collision probability function) and consequently enter through them. Second, non-front-line agents are able to anticipate the front-liners' continuing motion and align themselves behind them. In the resulting motion, agents re-organize themselves into lanes and are able to fluidly reach their destinations. This re-organization can be observed through the low number of emerging sub-groups (Figure 8(top)) which correspond to the formed lanes, and through the relatively low spread of the agents (Figure 8(bottom)).

In the case of the other algorithms, however, the groups can be observed to collide, block each other, and spread in order to allow agents (individual or in small groups) to pass through to their goals (a still of this can oberved on the left of Figure 7). For the ORCA algorithm, for example, this is due to the solution space quickly becoming saturated, thus forcing the agents to start spreading on the sides in order to free up the velocity space and be able to continue their motion. This disorganization can be observed through the high number of emerging sub-groups (Figure 8(top)) corresponding to agents searching for a less saturated solution space, thereby spreading over larger distances (Figure 8(bottom)).

Crossing In the Crossing situation (Figure 6), as expected agents simulated with our algorithm are able to cross without congestion (Figure 9, top: no jammed agents) forming the expected 45° crossing patterns (Figure 9, bottom).

Other algorithms' agents on the other hand, as can be seen on the right of Figure 7 quickly get into a congestion (Figure 9, top: increasing numbers of jammed agents) and no consistent patterns can be found, as seen on the bottom of Figure 9.

Summary Overall, WarpDriver is able to better find (and take advantage of) narrow spaces between agents (local minima in the collision probability fields) thus producing more visually pleasing results than the other algorithms, which often have more binary reactions, leading to entrapping agents in congested scenes.

7.2 Non-Linear Motion

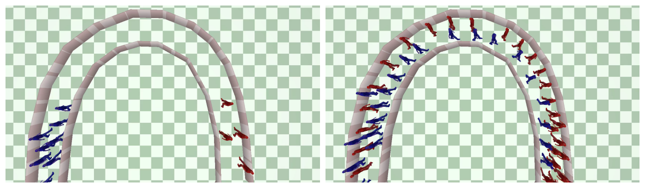


Figure 10: *Curved Flows example: two opposite flows of agents on a curved path (blue ones turn clockwise, red ones counter-clockwise). Simulated with WarpDriver.*

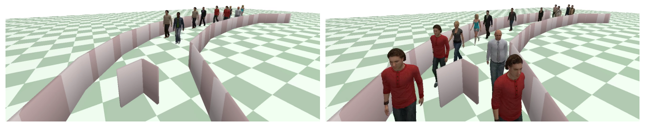


Figure 11: *Curved Obstacle example: a small obstacle is on the way of a flow of agents on a curved path. Simulated with WarpDriver.*

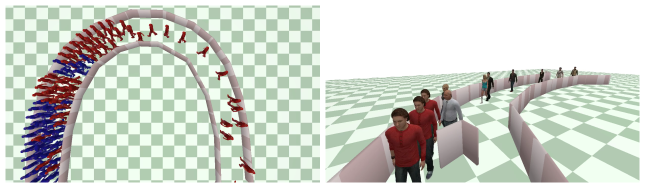


Figure 12: *Issues encountered in Curved Flows and Curved Obstacle. Left: Curved Flows, congestion observed for ORCA. Right: Crossing, Powerlaw agents can only pass the obstacle on their right.*

With the following test cases, we investigate how our algorithm copes with situations where agents' future motions are non-linear. To this end, we make agents interact with each other and with obstacles, while traveling along curved paths.

7.2.1 Description

Test case 3: Curved Flows In this situation (Figure 10), we set two opposing flows of agents (moderate density, about a meter between agents) in a curved corridor. Here, with the moderate density, we expect agents to fluidly navigate to the other end of the corridor.

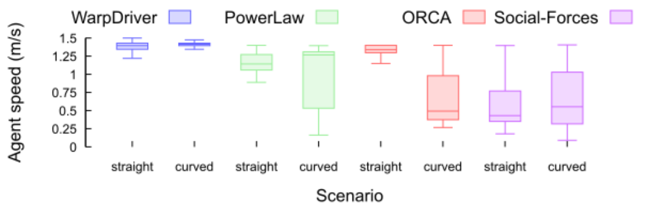


Figure 13: *Agent speeds in straight vs. curved corridors (same corridor length and width, same agent density). WarpDriver: consistent agent motions; other algorithms': important loss of speed in curved corridor. Full graphs in Appendix C.*

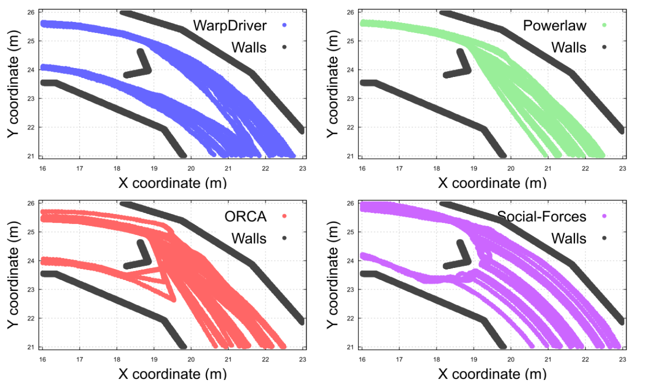


Figure 14: *Agent traces. WarpDriver agents pass the obstacle on the most convenient side. Social-Forces agents bump on the obstacle and pass on closest side. Powerlaw and ORCA agents can only pass on their right (some ORCA agents get pushed to left side by a small congestion).*

To measure the impact the curved corridor has on the agents, we reproduced the experiment in all aspects (same number and density of agents, same corridor length and width) except for one: we made the corridor straight. We then measured the average speed of the agents first in the straight version and then in the curved version, as seen in Figure 13.

Test case 4: Curved Obstacle This situation is a simplification of the previous test case: one uni-directional flow of agents is made to travel the same curved corridor with one small obstacle in the middle as shown on Figure 11. In this simple test case, we expect the agents to easily bypass the obstacle on the side that is most direct, i.e. if an agent is on the outer (resp. inner side) side of the corridor it should bypass the obstacle on the outer side (resp. inner side). We thus looked at the paths agents followed (Figure 14).

7.2.2 Analysis

Curved Flows In the Curved Flows example (Figure 10), with our algorithm agents are able to avoid each other correctly, with the emergence of a few opposing lanes facilitating flow. Furthermore, in Figure 13 we can see that using our algorithm, agents travel at the same overall speed in both the straight and curved versions.

With the other algorithms on the other hand, agents quickly get stuck in a congestion (Figure 12, left). We can observe this in Figure 13, which shows an important loss of agents' speed on the curved version of the corridor as compared to the straight one.

Curved Obstacle The Curved Obstacle situation shows the phenomenon more clearly. With our algorithm, agents anticipate the obstacle about 3m in advance (see Figure 14, top left) and choose the most direct (expected) side.

Agents from the Powerlaw and ORCA algorithms on the other hand can be seen to all prefer the outer side (with respect to the curve) of the obstacle (Figure 14, top right and bottom left) and some agents backtrack (Figure 14, top right) and use the inner side when a bottleneck situation forms. The Social-Forces algorithm produces results analogous to ours (Figure 14, bottom right): being position-based, this algorithm steers agents without anticipation and thus they "bump" on the obstacle and pass it on this same side.

Summary In both examples, the difference between our algorithm and the two velocity-based algorithms is that agents simulated with WarpDriver anticipate their own (and others') future trajectories as curved along the corridor, thus perceiving interactions where they most probably will occur (thus they see agents in the opposite flow and obstacle from the two examples well in advance). Velocity-based agents in these cases exhibit visual artifacts due to their linear extrapolation of trajectories based on instantaneous velocities: they can only perceive interactions that will occur roughly on a line tangent to the corridor curve at their position (thus they do not react in advance to agents in the opposite flow nor the obstacle from the two previous examples until the very last moment).

7.3 History-based Anticipation

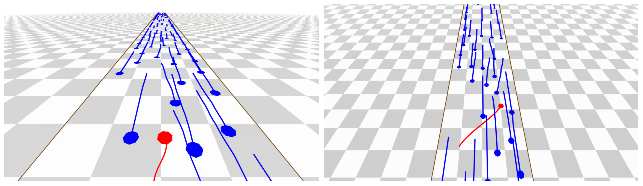


Figure 15: *Zig-Zags example: agents (blue) avoid a zig-zagging agent (red); left: narrow zig-zags, right: wide zig-zags. Simulated with WarpDriver.*

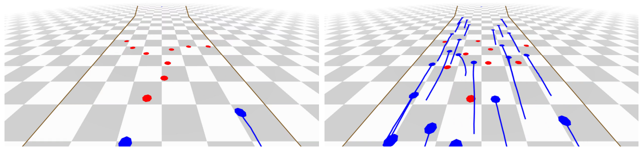


Figure 16: *Danger Corridor example: agents (blue) avoid turning obstacles (red). Simulated with WarpDriver.*

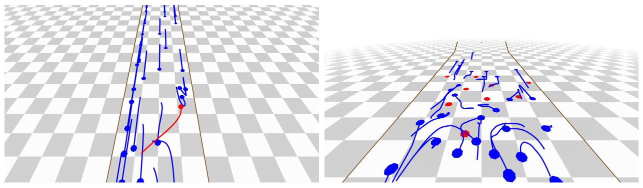


Figure 17: *Issues encountered in Zig-Zags and Danger Corridor. Left: Zig-Zags, powerlaw agents backtrack from zig-zagging agent. Right: Danger Corridor, ORCA agents backtracking and performing other erratic motions next to turning obstacles.*

As instantaneous velocities can vary very rapidly and not be representative of agents' overall motions, we next test situations where agents or obstacles behave according to pattern-like movements. We test two easily-recognizable behaviors: zig-zagging and revolving motions.

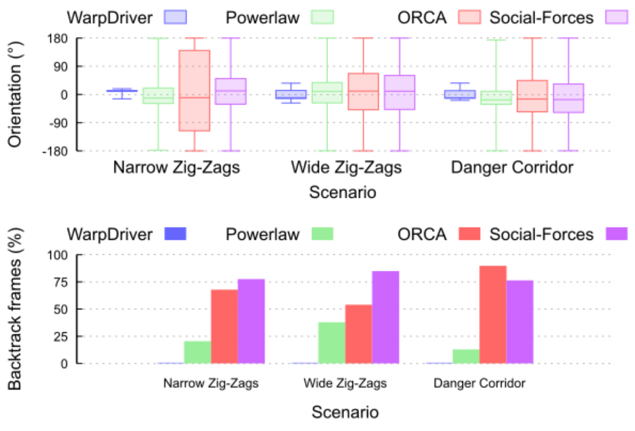


Figure 18: *Top: orientation of agents with respect to intended direction. WarpDriver agents are able to consistently head towards their intended direction. Other algorithms' agents make very strong adaptations to their motions. Full graphs in Appendix C. Bottom: percentage of simulated frames containing backtracking agents; No WarpDriver agent has been backtracking. Other algorithms contain many backtracking agents.*

7.3.1 Description

Test case 5: Zig-Zags In this scenario (Figure 15), we set up a uni-directional flow of moderately-spaced agents (in blue) traveling along a straight corridor and further add an agent (in red) which travels counter-flow with a zig-zagging trajectory. Figure 15 shows both cases where the red agent has a narrow zig-zagging motion (left) as well as a wide motion (right). In this example, we expect the blue agents to recognize and anticipate the red one's motion pattern and easily avoid it. We measured how easily blue agents are able to avoid the red one by recording the angle between the agents' orientation and their goal direction (their deviation from their goal) on Figure 18(top). We also report what proportion of the simulated frames contain backtracking agents (180° deviations) on Figure 18(bottom).

Test case 6: Danger Corridor This scenario (Figure 16) is largely similar to the previous one in that a uni-directional flow of agents (in blue) travel down a corridor, except that we set nine slowly revolving pillars (in red) in the middle of the path. We then expect agents to be able to recognize how these pillars move and easily work out a path through them. Again, we measure the agents' deviation from their goals which we report on Figure 18(top) and the proportion of frames containing backtracking agents on Figure 18(bottom).

7.3.2 Analysis

Zig-Zags In the Zig-Zag examples (Figure 15), agents (in blue) simulated with WarpDriver are able to anticipate the zig-zagging agent (in red) in advance and minimally adapt their trajectories to avoid it. This is confirmed by Figure 18(top) which shows that the heading direction of the agents is very close to 0° (heading in their preferred direction).

Other algorithms' agents, on the other hand, have more trouble anticipating the jerky motion of the zig-zagging agent and noticeably over-react as a result. This is confirmed by the large spreads of boxplots from Figure 18(top) where agents often deviate by ±180° (i.e. backtracking from their goal, as seen on Figure 18(bottom)).

Danger Corridor The Danger Corridor example (Figure 16) yields results largely similar to the Zig-Zags one (but more pronounced). WarpDriver agents are able to fluidly avoid the revolving obstacles with similarly little deviation (as for the Zig-Zags) from their goal direction (Figure 18(top)).

Other agents have, again, more trouble dealing with the situation, with much larger deviations from their intended directions (Figure 18(top)), and a noticeable amount of backtracking agents (Figure 18(bottom)). Non-similarly to the Zig-Zags however, in the case of the Danger Corridor, the Powerlaw algorithm produces many collisions with the revolving obstacles (a collision is found when the center of an obstacle is inside the radius of an agent; 21% of the simulation frames contained collisions for Powerlaw, compared to less than 0.5% for ORCA/Social-Forces and 0% for WarpDriver).

Summary Overall, the differences can be explained by the fact that in these cases, the instantaneous velocities of the zig-zagging agent and revolving obstacles are constantly changing and their trajectories are not straight. Thus, velocity-based algorithms first linearly extrapolate (incorrect) future motions and then face these extrapolations constantly changing. The resulting agents thus avoid many, ever-changing and possibly non-existent future interactions, with large deviations from their intended directions and many agents backtracking away from their goal.

These artifacts are addressed by WarpDriver: first, it detects patterns in the past motions and learns from them to anticipate future motions; second, when anticipating future motions it does so non-linearly. Thus, WarpDriver agents are able to correctly anticipate and avoid collisions, resulting in more natural reactions (low deviation from intended directions), and no back-tracking. Although here all agents are given the same preferred speed (for legibility on video, and not to affect metrics), the quality of results remains the same with speed variations (see supplemental video clips).

7.4 Highly-Constrained Space

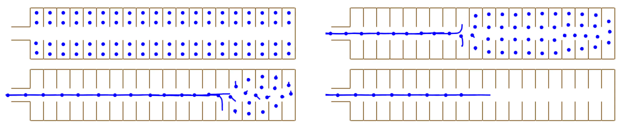


Figure 19: *Plane example: plane exit situation involving 80 agents. Simulated with WarpDriver.*

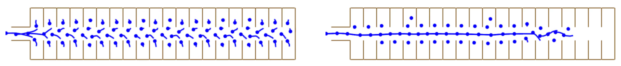


Figure 20: *Issues encountered in Plane: Powerlaw agents from aisle seats exit before everyone else.*

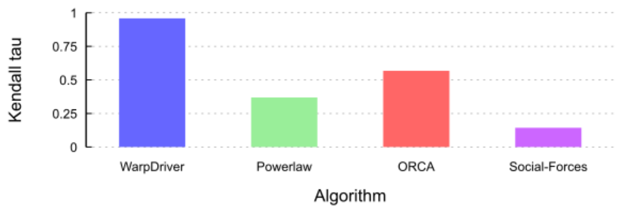


Figure 21: *Kendall tau coefficient for plane exit order, higher is better. WarpDriver produces an order very close to the expected one, while other algorithms produce much further orders.*

In the last scenario, we test our algorithm on coping with highly-constrained scenarios, as in very confined spaces, where agents are within contact distance and many encounter path intersections.

7.4.1 Test case 7: Plane

This example features a plane egress scenario with 80 agents (Figure 1 and Figure 19). Here, we expect agents to orderly exit the plane starting with the ones close to the exit and with more far-away agents exiting last. To see how agents are able to cope with this situation, we assigned to each agent the number of its row (e.g. the four agents of the first row have the number 1, the four agents of the last row have the number 20), then we recorded the number sequence of agents as they got out of the plane and compared it using the Kendall tau measure [Kendall 1938] to the ideal exit sequence $[1, 1, 1, 1, ..., 20, 20, 20, 20]$ (Figure 21).

7.4.2 Analysis

As can be seen on Figure 19, with our algorithm, agents in the back allow agents up front to exit first. This leads to an orderly exiting process, where all agents are progressively evacuated, as evidenced by the high Kendall tau coefficient (0.96) which indicates a close to ideal exit sequence Figure 21. The behavior obtained with our algorithm results from a combination of factors. First, agents are able to predict which way the others will go: into the alley and then towards the exit (note that these paths are non-linear since they contain a right turn). Second, agents in the front rows are closer to the exit than the others and they are thus perceived as obstacles blocking the exit from the other agents (and conversely front agents perceive other agents as being "behind"), thus creating a hierarchy. Finally, the agents easily navigate between the chairs by following the local minima of the collision probabilities defined by these obstacles.

On the contrary with the other algorithms, all agents try to exit at the same time which, with the very constrained space (little room for maneuvers) leads to unorderly behaviors. For instance, in the case of the Powerlaw algorithm (Figure 20), aisle agents all gather in the alley at the same time and exit before everyone else; while the alley agents exit, window agents from the back have more space and exit next; overall, window agents from the front and middle rows are last to exit. This general lack of order is further confirmed by the much lower Kendall tau values for Powerlaw, ORCA and Social-Forces found in Figure 21. In order for these algorithms' agents to exit in order, additional scripting would be required.

7.5 Benchmarks

Previous results provide a quantitative evaluation of visual artifacts, including comparisons with previous techniques; this section provides additional evaluation of results along two aspects: comparisons with real data and an analysis of algorithmic complexity and computational performance.

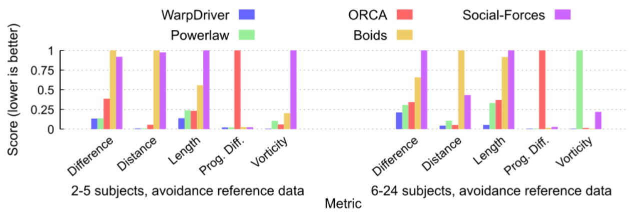


Figure 22: *Benchmarks results using the method from [Wolinski et al. 2014], lower is better.*

Data-driven validation Finally, we compare our algorithm's performance with the Powerlaw, ORCA, Social-Forces, and Boids algorithms on previously-studied test cases using the method from [Wolinski et al. 2014]. In these tests, the difference between our algorithm and the others is not always as pronounced as in the previously shown scenarios. This is due to the nature of the available ground truth data which only captures simple interactions: (1) simple crossing situations between 2-5 agents, and (2) 6-24 agents exchanging positions on a circle. Nonetheless, as Figure 22 shows, on these test cases, our method (in blue) gives comparable results to velocity-based algorithms (Powerlaw - green, and ORCA - red), occasionally outperforming them (and almost always outperforming the other algorithms).

Complexity Like for most other simulation algorithms, the base complexity of our approach is quadratic, $O(n^2)$: every agent interacts with every other agent. Most algorithms deal with this using space-optimization structures such as kd-trees that reduce the runtime complexity by limiting the number of neighbors for a given agent, but with the possible risk of arbitrarily discarding important agents and thereby degrading results.

While we could also use such a strategy for WarpDriver, we note that it is algorithmically very close to ray-tracing. We can thus borrow strategies from the wide associated literature, such as parallel sampling, caching, level-of-detail, etc. We have implemented one such strategy, where in a pre-processing phase at the start of each timestep, each agent imprints a theoretical maximum bounding volume of its associated collision probability field onto a grid. Then, when an agent samples collision probabilities, instead of sampling every other agent's field, it only samples the fields of those that have their ID imprinted at that location on the grid. As a crowd is not infinitely compressible, there is a maximum number of interactable neighboring agents, thus giving our algorithm a linear upper bound to its complexity of $O(n)$. This technique allows us to have the same simulations with and without it: i.e. we can optimize the runtime complexity without degrading the simulation results (though it would be interesting to study the effects of limiting neighbors).

In practice, assuming the typical target 15-20 fps framerate for the motion of crowds, our algorithm can simulate $5,000$ agents in real time. In comparison, on the same machine and for the same number of agents, ORCA runs at $\sim$140 fps. Powerlaw on the other hand, for stability reasons requires much lower timesteps – values of $\sim$0.005s can be found in the examples bundled with the source code, which means it needs to run at 200 fps to be real-time – and falls to $\sim$40 fps on the *Big Groups* example that involves 2054 agents.

8 Discussion and Limitations

This probabilistic motion prediction algorithm for crowd simulation accounts for the contextual interaction between the agent and its surroundings, including other agents, the environment layout, past motions, etc. We assume that environments can be annotated with probable routes to be followed by agents. This does not present any difficulty – it just needs to be done once for each new environment, and could easily be automated. Probable routes' geometry could be extracted automatically based on smoothed Voronoï diagrams (or any other technique computing static obstacles' medial axes), or even learned from real data (e.g. camera feeds).

Although in this paper we have focused on the application of motion prediction to collision avoidance for crowd simulation, motion prediction is generally the core of numerous types of interactions among agents and it represents the most basic software module of all crowd simulators. Thus, our method can and should be easily extended to handle other forms of interactions, including following, fleeing, intercepting, group behaviors, etc.

A possible limitation is that the current formulation does not distinguish collision sources. For instance, equivalent collision probabilities between a neighbor agent moving in the same direction and one moving in the opposite direction are processed the same way. They, however, do not result in the same energy of collision, which could be integrated into the notion of "risk of collision". Theoretically, our method can handle any kind of moving obstacle, and extending it with this notion of risk would allow us to mix into our simulations other types of agents (e.g. cars) with their corresponding level of danger. Finally, in extreme cases, a single step of gradient descent per simulation timestep might not be sufficient. Additional iterations or heuristics could be useful.

9 Conclusion

In this paper, we present a new context-aware motion prediction algorithm for crowd simulation, with two main results.

First, given its non-deterministic and probabilistic representation for motion prediction, agents no longer perceive future collisions in a binary manner as in most existing methods, but rather as continuous probability fields. This offers several advantages: (1) Due to the continuity of the probability fields, motions are smoother; agents lower the probabilities of colliding by following probability gradients. (2) Agent oscillations between two binary future collision states often observed in some previous techniques are avoided. (3) Multiple possible hypotheses can be maintained simultaneously, the notion of routes can be used when future motion probabilities are propagated in time. (4) The non-determinism allows us to simulate uncertainties due to sensing or variation in locomotion trajectories. By increasing the uncertainty of agents' future positions the further they are in time, we change the relative importance of agents that may collide sooner/later. (5) Even if *Warp Operators* implement conflicting heuristics, as long as they produce justifiable collision probabilities, the trajectory adaptation mechanism (currently gradient descent) will find an acceptable motion.

The second innovation is related the contextual awareness of our technique, which not only considers agents' states, but also external/contextual cues. This insight introduces a major departure from previous methods that assume agents keep moving with the same current velocity vector. One can easily conceive that agents' current velocity vectors may not be representative of the intention of future motion, especially in dense crowds, where agents are constantly adapting their locomotion trajectories to the presence of others.

Through a set of challenging benchmark scenarios and quantitative evaluations, we demonstrate that this new probabilistic theoretic framework for motion prediction considerably improves the visual quality of crowd simulations, alleviating artifacts commonly observed in some state-of-the-art collision-avoidance algorithms.

Addressing each of the limitations mentioned above can lead to promising directions for future work. While we have presented noticeable improvements in terms of agent motion quality, investigating each of these new aspects would likely result in next-generation crowd simulators capable of matching real observations more accurately in the near term. Furthermore, we would like to adapt our simulator to consider other forms of local interactions in addition to collision avoidance. One promising direction is to learn future position likelihoods based on real observations, allowing to automatically adapt our simulator to a specific situation. In a given place, the probability of future positions depends on the nature of people who frequent this specific place, and on the exact activities they perform there. Without the need to explicitly specify this knowledge, we could easily learn the resulting probability fields.

Acknowledgments

This research is supported in part by the French National Research Agency (PERCOLATION project ANR-13-JS02-0008), the Inria Associated Team SIMS, and the U.S. Army Research Office.

References

CHENNEY, S. 2004. Flow tiles. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, 233–242.

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

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

GOLAS, A., NARAIN, R., AND LIN, M. 2013. Hybrid long-range collision avoidance for crowd simulation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, I3D '13, 29–36.

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 *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 177–187.

GUY, S. J., CURTIS, S., LIN, M. C., AND MANOCHA, D. 2012. Least-effort trajectories lead to emergent crowd behaviors. *Phys. Rev. E 85* (Jan), 016110.

GUY, S. J., VAN DEN BERG, J., LIU, W., LAU, R., LIN, M. C., AND MANOCHA, D. 2012. A statistical similarity measure for aggregate crowd dynamics. *ACM Trans. Graph. 31*.

HELBING, D., AND MOLNÁR, P. 1995. Social force model for pedestrian dynamics. *Physical Review E 51*, 5, 4282–4286.

HELBING, D., FARKAS, I., AND VICSEK, T. 2000. Simulating dynamical features of escape panic. *Nature 407*, 6803, 487–490.

JIN, X., XU, J., WANG, C. C. L., HUANG, S., AND ZHANG, J. 2008. Interactive control of large-crowd navigation in virtual environments using vector fields. *IEEE Comput. Graph. Appl. 28*, 6 (Nov.), 37–46.

JU, E., CHOI, M., PARK, M., LEE, J., LEE, K., AND TAKAHASHI, S. 2010. Morphable crowds. *ACM Trans. Graph. 29*.

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

KARAMOUZAS, I., HEIL, P., BEEK, P., AND OVERMARS, M. H. 2009. A predictive collision avoidance model for pedestrian simulation. In *Proceedings of the 2nd International Workshop on Motion in Games*, Springer-Verlag, Berlin, Heidelberg, 41–52.

KARAMOUZAS, I., SKINNER, B., AND GUY, S. J. 2014. Universal power law governing pedestrian interactions. *Phys. Rev. Lett. 113* (Dec), 238701.

KENDALL, M. G. 1938. A new measure of rank correlation. *Biometrika 30*, 1/2, 81–93.

KIM, S., GUY, S. J., LIU, W., WILKIE, D., LAU, R. W., LIN, M. C., AND MANOCHA, D. 2014. Brvo: Predicting pedestrian trajectories using velocity-space reasoning. *The International Journal of Robotics Research*.

KRETZ, T., AND SCHRECKENBERG, M. 2008. The f.a.s.t.-model. *CoRR abs/0804.1893*.

LERNER, A., CHRYSANTHOU, Y., AND LISCHINSKI, D. 2007. Crowds by example. *Computer Graphics Forum 26*, 3, 655–664.

LIU, C. K., HERTZMANN, A., AND POPOVIĆ, Z. 2005. Learning physics-based motion style with nonlinear inverse optimization. *ACM Trans. Graph. 24*, 3 (July), 1071–1081.

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

OLIVIER, A.-H., MARIN, A., CRÉTUAL, A., AND PETTRÉ, J. 2012. Minimal predicted distance: A common metric for collision avoidance during pairwise interactions between walkers. *Gait & posture 36*, 3, 399–404.

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

PARIS, S., PETTR, J., AND DONIKIAN, S. 2007. Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum 26*, 3, 665–674.

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

PELLEGRINI, S., ESS, A., SCHINDLER, K., AND VAN GOOL, L. 2009. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Computer Vision, 2009 IEEE 12th International Conference on*, 261–268.

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 *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, NY, USA, 189–198.

REYNOLDS, C. W. 1987. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Computer Graphics 21*, 25–34.

REYNOLDS, C. 1999. Steering behaviors for autonomous characters. In *Game Developers Conference 1999*, 763–782.

SCHADSCHNEIDER, A. 2001. Cellular automaton approach to pedestrian dynamics - theory. 11.

TREUILLE, A., COOPER, S., AND POPOVIĆ, Z. 2006. Continuum crowds. In *SIGGRAPH '06*, ACM, NY, USA, 1160–1168.

VAN DEN BERG, J., LIN, M., AND MANOCHA, D. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *IEEE International Conference on Robotics and Automation*.

VAN DEN BERG, J., SNAPE, J., GUY, S., AND MANOCHA, D. 2011. Reciprocal collision avoidance with acceleration-velocity obstacles. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 3475–3482.

VAN DEN BERG, J., GUY, S. J., LIN, M., AND MANOCHA, D. 2011. Reciprocal n-body collision avoidance. In *Robotics Research*. Springer, 3–19.

WOLINSKI, D., GUY, S., OLIVIER, A.-H., LIN, M., MANOCHA, D., AND PETTRÉ, J. 2014. Parameter Estimation and Comparative Evaluation of Crowd Simulations. *Computer Graphics Forum 33*, 2, 303–312.

ZHOU, B., TANG, X., AND WANG, X. 2012. Coherent filtering: detecting coherent motions from crowd clutters. In *Computer Vision–ECCV 2012*. Springer, 857–871.