



HAL
open science

WarpDriver: Context-Aware Probabilistic Motion Prediction for Crowd Simulation

David Wolinski, Ming C. Lin, Julien Pettré

► **To cite this version:**

David Wolinski, Ming C. Lin, Julien Pettré. WarpDriver: Context-Aware Probabilistic Motion Prediction for Crowd Simulation. ACM Transactions on Graphics, 2016, 35 (6), 10.1145/2980179.2982442 . hal-01411087

HAL Id: hal-01411087

<https://inria.hal.science/hal-01411087>

Submitted on 7 Dec 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

WarpDriver: Context-Aware Probabilistic Motion Prediction for Crowd Simulation

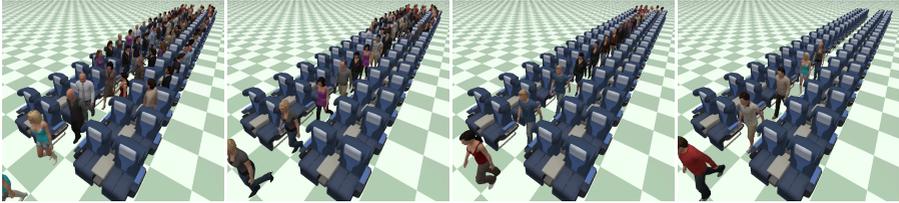


Figure 1: WarpDriver agents exiting a plane in seating order, solely due to collision avoidance, without additional scripting.

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.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation;

Keywords: crowd simulation, anticipation, collision avoidance

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 the-

oretic 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 [Pellegriani 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 [Kapidia 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).

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 collision 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.

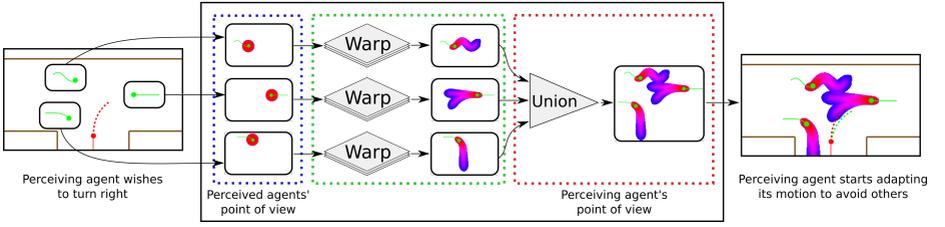


Figure 2: Overview of the algorithmic framework of WarpDriver.

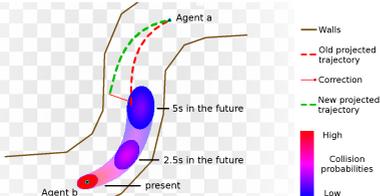


Figure 3: Illustration of collision avoidance between two agents a and b on a curved path. The color gradient represents a 's probability of collision with b , as perceived by a . The red dotted line represents a 's initial projected trajectory. The green dotted line represents a 's final, corrected, 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.

Our goal for the collision probability formulation process (Step 2) is to be able to handle each *property* separately. Thus, we define the *Intrinsic Field* as the lowest common denominator among all 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 be simply defined as a line in its *agent-centric* space-time, which simplifies further computations (more detail in Sections 4, 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.
- $\vec{\nabla}$ is the nabla operator. For a continuous field f , $\vec{\nabla} \cdot f$ is the gradient of f .
- \cup is the union operator and \bigcup is the union operator over a set.
- \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 the space of 2D positions and \mathbf{t} is the time. A point in such a space-time is noted $\mathbf{s} = (x, y, t) \in \mathcal{S}$. 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.
- $\mathbf{r}_{a,k}$ is agent a 's *projected trajectory* in $\mathcal{S}_{a,k}$.
- $\forall \mathbf{s} \in \mathcal{S}_{a,k}$, $p_{a \rightarrow b,k}(\mathbf{s})$ 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 $\mathcal{S}_{b,k}$. $\vec{\nabla} \cdot I$ is the gradient of I .
- W denotes a *Warp Operator* that warps I for every *property* of an agent. $\mathbf{W} = W_n \circ \dots \circ W_1$ further denotes the composition of operators $\{W_1, \dots, W_n\}$.
- W^{-1} is used to apply the inverse of a *Warp Operator* W to probabilities and probability gradients. Assuming a collection of operators $\{W_1, \dots, W_n\}$ where $\mathbf{W}(\mathcal{S}_{a,k}) = \mathcal{S}_{b,k}$, then $\mathbf{W}^{-1} = W_1^{-1} \circ \dots \circ W_n^{-1}$ and $\forall \mathbf{s} \in \mathcal{S}_{a,k}$:

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

$$(\mathbf{W}^{-1} \circ (\vec{\nabla} \cdot I) \circ \mathbf{W})(\mathbf{s}) = \vec{\nabla} \cdot p_{a \rightarrow b,k}(\mathbf{s}). \quad (2)$$

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 expressed as $\mathbf{r}_{a,k} = \text{line}(\mathbf{o}, v_{a,k}\mathbf{x} + \mathbf{t})$. Further, 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 :

$$\forall \mathbf{s} = (x, y, t) \in \mathcal{S}_{b,k}, g(\mathbf{s}) = \begin{cases} 1, & \text{if } \sqrt{x^2 + y^2} \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

We further model the perception error in the form of a Gaussian function: $\forall \mathbf{s} = (x, y, t) \in \mathcal{S}_{b,k}, f(\mathbf{s}) = \exp(-(\frac{x^2+y^2}{2\sigma^2}))$.

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

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

It is computed up to a normalized time of 1 second in the future. An illustration of the *Intrinsic Field* can be found on Figure 2 (cylinder on the right side of the figure).

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*. Note that their formal expressions are given 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 x and y axes.

Velocity The W_v operator models the agent's instantaneous velocity as a displacement along the 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 than 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: $W_{ref} = \{W_{local} \text{ or } W_{el} \text{ or } W_{io} \text{ or } W_{ob}\}$.

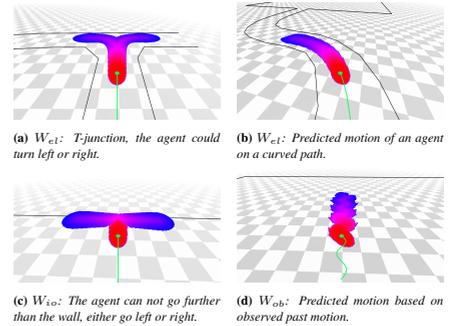


Figure 4: Cases using context-related Warp Operators (Section 5.2.2). Each case represents one context-related Warp Operator combined with all agent-related ones (Section 5.2.1). Same simplified 2D representation as in Figure 3(right).

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,

345 have a tendency to stay roughly in the middle, resulting in a curved
 346 trajectory (Figure 4b). The operator W_{el} models this knowledge by
 347 warping space to “align” it with these probable trajectories.

348 **Interactions With Obstacles** Where the environment layout oper-
 349 ator focuses on other agents’ probable trajectories assuming they
 350 will continue travelling, this operator W_{io} takes care of possible
 351 interactions between agents and obstacles. These interactions are
 352 essentially much more drastic changes to an agent’s locomotion
 353 than paths, such as full stops. These can occur if, for instance,
 354 an agent comes up to a wall (Figure 4c) (to interact with an ATM,
 355 look out the window, check a map...). This can also happen with an
 356 agent coming into contact with a small/temporary/unexpected ob-
 357 stacle which could force it to stop and then “hug” the obstacle to
 358 get around it.

359 To achieve this, we construct a graph around each obstacle (an ob-
 360 stacle being modeled as a series of connected line segments). When
 361 an agent’s *projected trajectory* intersects with an obstacle, we ex-
 362 tend the graph to that agent and “align” space-time with this graph.

363 **Observed Behaviors** With the last operator W_{ob} , we aim to im-
 364 prove the prediction of agents’ future motions by looking at their
 365 past ones. In the worst case, we might not find any useful infor-
 366 mation, which won’t impact the prediction. However, we might also
 367 find some behaviors similar to what the agent is currently doing
 368 (e.g. turning in a particular way) or, in the best case, we might find
 369 patterns (e.g. agents going in near-circles, zig-zags...) that we can
 370 extend to the currently-observed situation (Figure 4d shows antici-
 371 pation on a zig-zagging agent).

372 In order to take this information into account for an agent a at
 373 timestep k , we keep a history of this agent’s positions during h previ-
 374 ous timesteps. These past positions form a graph which we repeat
 375 on the current position of the agent and then “align” space-time
 376 with it.

377 5.2.3 Composition of Warp Operators

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

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

For any point \mathbf{s} in *perceiving agent a’s agent-centric* space-time
 $S_{a,k}$:

$$\begin{aligned} p_{a \rightarrow b,k}(\mathbf{s}) &= (\mathbf{W}^{-1} \circ I \circ \mathbf{W})(\mathbf{s}), \\ \vec{\nabla} \cdot p_{a \rightarrow b,k}(\mathbf{s}) &= (\mathbf{W}^{-1} \circ (\vec{\nabla} \cdot I) \circ \mathbf{W})(\mathbf{s}). \end{aligned}$$

378 5.3 Combining collision probability Fields

Before the collision avoidance problem can be solved, one last me-
 403 chanic still needs to be defined which is how pair-wise interactions
 404 can be combined (Step 3 on Figure 2). Let a be the *perceiving*
 405 agent, and $b, c \in \mathcal{A}$, $b \neq a$, $c \neq a$ be a pair of *perceived* agents. At
 406 timestep k , we have access to the following collision probabilities:
 407 $p_{a \rightarrow b,k}$ and $p_{a \rightarrow c,k}$. We can then define the probability agent a has
 408 of colliding with either b or c :

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

And we can similarly define its gradient:

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

379 Finally, considering the whole set of agents \mathcal{A} , the probability agent
 380 a has of colliding with any other agent $b \in \mathcal{A} \setminus a$ is obtained in the
 381 same manner and noted $p_{a \rightarrow \mathcal{A} \setminus a,k}$, with the corresponding gradient
 382 $\vec{\nabla} \cdot p_{a \rightarrow \mathcal{A} \setminus a,k}$.

383 6 Solving the Collision-Avoidance Problem

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

387 To solve the collision-avoidance problem, the *perceiving* agent
 388 samples collision probabilities and their gradients $p_{a,k}$ along its
 389 *projected trajectory* $\mathbf{r}_{a,k}$ (this is the cost function and its gradient),
 390 and performs one step of gradient descent to modify its *projected*
 391 *trajectory*. First, we compute the overall probability an agent a has
 392 of colliding with other agents $p_{a,k}$, its gradient $\vec{\nabla} \cdot p_{a,k}$ and applica-
 393 tion point $\mathbf{s}_{a,k}$ (red continuous line in Figure 3), when traveling
 394 along its *projected trajectory* $\mathbf{r}_{a,k}$ (red dotted curve in Figure 3).
 395 We compute these quantities for a time horizon \mathcal{T}^* until a collision
 396 with a wall is detected: $\mathcal{T}^* \leq \mathcal{T}$. With the following normalization
 397 factor $N_{a,k}$, and $t \in [0, \mathcal{T}^*]$:

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

We compute $p_{a,k}$, $\vec{\nabla} \cdot p_{a,k}$ and $\mathbf{s}_{a,k}$:

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

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

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

388 From these quantities, given a user-set parameter α , we compute
 389 the new projected trajectory that agent a should follow in $S_{a,k}$ to
 400 lower its collision probability (green dotted curve in Figure 3):

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

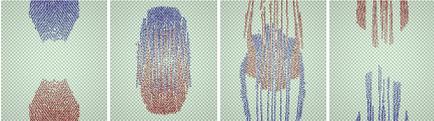
401 7 Results

402 In this section, we show the benefits of our WarpDriver algo-
 403 rithm as compared with several existing methods. To illustrate
 404 the advantages of the more complex *Warp Operators*, we compare
 405 WarpDriver with two velocity-based algorithms: the well-known
 406 ORCA algorithm [Van Den Berg et al. 2011b] and the recent Pow-
 407 erlaw algorithm [Karamouzas et al. 2014], as they are representa-
 408 tive of what can be achieved with velocity-based approaches. We
 409 also compare WarpDriver with two position-based algorithms: the
 410 Boids [Reynolds 1987] and the Social-Forces [Helbing and Molnár
 411 1995] algorithms.

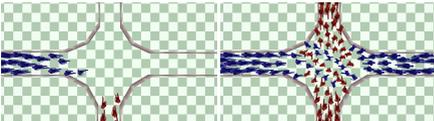
412 First, we test WarpDriver in challenging scenarios, including large,
 413 dense crowds, scenarios with non-linear routes, history-based
 414 anticipation cases and a highly-constrained situation. We show the
 415 results of our algorithm vs. Powerlaw, ORCA, and Social-Forces
 416 (Boids is omitted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

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

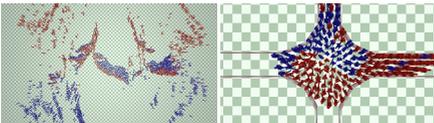
424 7.1 Large and Dense Crowds



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



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



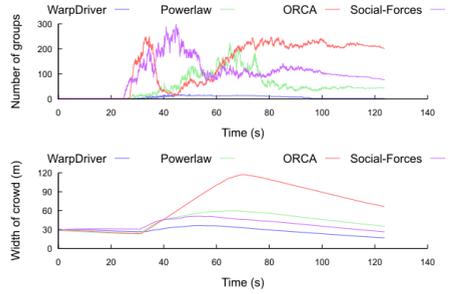
430 **Figure 7:** Issues encountered in Big Groups and Crossing. Left:
 431 Big Groups, ORCA agents block each other. Right: Crossing, con-
 432 gestion observed for Powerlaw.

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

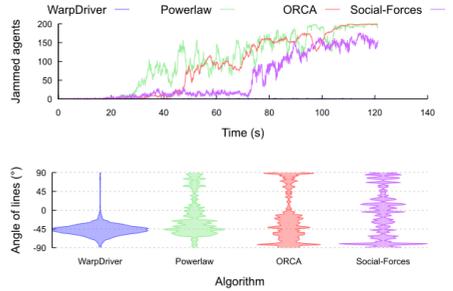
437 7.1.1 Description

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

446 **Test case 2: Crossing** The second test case involves two corri-
 447 dors intersecting at a right angle, each with a uni-directional flow



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



452 **Figure 9:** Top: number of jammed agents, close to none for Warp-
 453 Driver, many for other algorithms. Bottom: violin plots of detected
 454 agent lines at crossing intersection, consistently around -45° for
 455 WarpDriver, scattered (and fewer detected lines) for other algo-
 456 rithms; full graphs in Appendix B.

440 of agents (Figure 6). This kind of situation is well studied and
 441 45° lines should form between agents of each flow at the intersection
 442 [Cividini et al. 2013], facilitating their movement. We measure
 443 this by detecting sub-groups with the previously mentioned method
 444 and perform linear regression on the agents, results are reported
 445 on Figure 9(middle, bottom). Furthermore, as the situation is very
 446 constrained (agents at contact distance from each other with the
 447 presence of walls), we also measure how many agents are jammed
 448 (travel at less than 0.1m/s) during the simulations, as shown in Fig-
 449 ure 9(top).

450 7.1.2 Analysis

451 **Big Groups** In the Big Groups example (Figure 5), agents simu-
 452 lated with our algorithm are able to do two things. First, front-
 453 line agents are able to find points of entry in the opposing group
 454 (which correspond to the minima of the collision probability func-
 455 tion) and consequently enter through them. Second, non-front-line
 456 agents are able to anticipate the front-liners’ continuing motion and
 457 align themselves behind them. In the resulting motion, agents re-
 458 organize themselves into lanes and are able to fluidly reach their
 459 destinations. This re-organization can be observed through the low

460 number of emerging sub-groups (Figure 8(top)) which correspond
 461 to the formed lanes, and through the relatively low spread of the
 462 agents (Figure 8(bottom)).

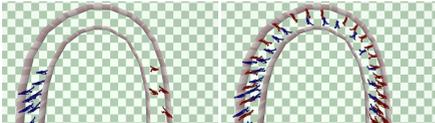
463 In the case of the other algorithms, however, the groups can be ob-
 464 served to collide, block each other, and spread in order to allow
 465 agents (individual or in small groups) to pass through to their goals
 466 (a still of this can be observed on the left of Figure 7). For the ORCA
 467 algorithm,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

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

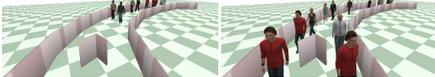
478 Other algorithms' agents on the other hand, as can be seen on the
 479 right of Figure 7 quickly get into a congestion (Figure 9, top: in-
 480 creasing numbers of jammed agents) and no consistent patterns can
 481 be found, as seen on the bottom of Figure 9.

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

487 **7.2 Non-Linear Motion**

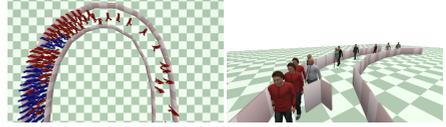


490 **Figure 10:** Curved Flows example: two opposite flows of agents
 491 on a curved path (blue ones turn clockwise, red ones counter-
 492 clockwise). Simulated with WarpDriver.

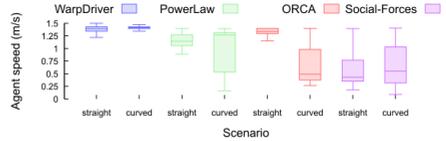


493 **Figure 11:** Curved Obstacle example: a small obstacle is on the
 494 way of a flow of agents on a curved path. Simulated with War-
 495 pDriver.

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



496 **Figure 12:** Issues encountered in Curved Flows and Curved Ob-
 497 stacle. Left: Curved Flows, congestion observed for ORCA. Right:
 498 Crossing, Powerlaw agents can only pass the obstacle on their
 499 right.

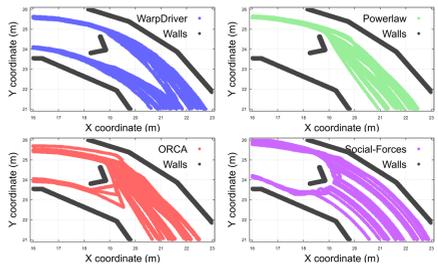


500 **Figure 13:** Agent speeds in straight vs. curved corridors (same
 501 corridor length and width, same agent density). WarpDriver: con-
 502 sistent agent motions; other algorithms': important loss of speed in
 503 curved corridor. Full graphs in Appendix B.

504 **7.2.1 Description**

505 **Test case 3: Curved Flows** In this situation (Figure 10), we set
 506 two opposing flows of agents (moderate density, about a meter be-
 507 tween agents) in a curved corridor. Here, with the moderate density,
 508 we expect agents to fluidly navigate to the other end of the corridor.
 509 To measure the impact the curved corridor has on the agents, we re-
 510 produced the experiment in all aspects (same number and density of
 511 agents, same corridor length and width) except for one: we made the
 512 corridor straight. We then measured the average speed of the agents
 513 first in the straight version and then in the curved version, as seen
 514 in Figure 13.

515 **Test case 4: Curved Obstacle** This situation is a simplification
 516 of the previous test case: one uni-directional flow of agents is made
 517 to travel the same curved corridor with one small obstacle in the
 518 middle as shown on Figure 11. In this simple test case, we expect



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

507 the agents to easily bypass the obstacle on the side that is most
 508 direct, i.e. if an agent is on the outer (resp. inner side) of the
 509 corridor it should bypass the obstacle on the outer side (resp. inner
 510 side). We thus looked at the paths agents followed (Figure 14).

511 7.2.2 Analysis

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

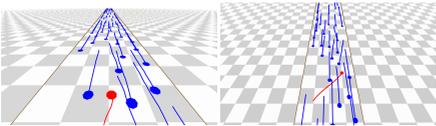
518 With the other algorithms on the other hand, agents quickly get
 519 stuck in a congestion (Figure 12, left). We can observe this in
 520 Figure 13, which shows an important loss of agents' speed on the
 521 curved version of the corridor as compared to the straight one.

522 **Curved Obstacle** The Curved Obstacle situation shows the phe-
 523 nomenon more clearly. With our algorithm, agents anticipate the
 524 obstacle about 3m in advance (see Figure 14, top left) and choose
 525 the most direct (expected) side.

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

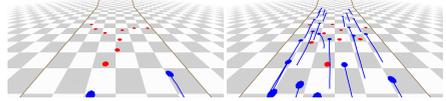
534 **Summary** In both examples, the difference between our algo-
 535 rithm and the two velocity-based algorithms is that agents simu-
 536 lated with WarpDriver anticipate their own (and others') future tra-
 537 jectories as curved along the corridor, thus perceiving interactions
 538 where they most probably will occur (thus they see agents in the op-
 539 posite flow and obstacle from the two examples well in advance).
 540 Velocity-based agents in these cases exhibit visual artifacts due to
 541 their linear extrapolation of trajectories based on instantaneous ve-
 542 locities: they can only perceive interactions that will occur roughly
 543 on a line tangent to the corridor curve at their position (thus they do
 544 not react in advance to agents in the opposite flow nor the obstacle
 545 from the two previous examples until the very last moment).

546 7.3 History-based Anticipation

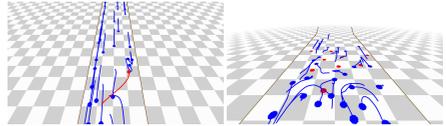


547 **Figure 15:** Zig-Zags example: agents (blue) avoid a zig-zagging
 548 agent (red); left: narrow zig-zags, right: wide zig-zags. Simulated
 549 with WarpDriver.

547 As instantaneous velocities can vary very rapidly and not be repre-
 548 sentative of agents' overall motions, we next test situations where
 549 agents or obstacles behave according to pattern-like movements.
 550 We test two easily-recognizable behaviors: zig-zagging and revol-
 551 ving motions.



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



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

558 7.3.1 Description

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

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

575 7.3.2 Analysis

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

582 Other algorithms' agents, on the other hand, have more trouble anti-
 583 cipating the jerky motion of the zig-zagging agent and noticeably
 584 over-react as a result. This is confirmed by the large spreads of
 585 boxplots from Figure 18(top) where agents often deviate by $\pm 180^\circ$
 586 (i.e. backtracking from their goal, as seen on Figure 18(bottom)).

587 **Danger Corridor** The Danger Corridor example (Figure 16)
 588 yields results largely similar to the Zig-Zags one (but more pro-
 589 nounced). WarpDriver agents are able to fluidly avoid the revolving
 590 obstacles with similarly little deviation (as for the Zig-Zags) from

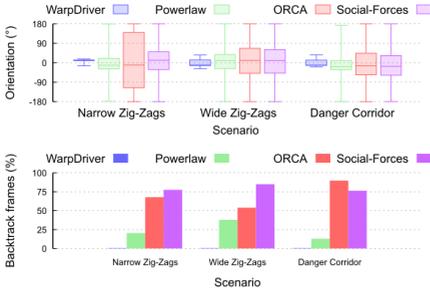


Figure 18: Top: orientation of agents with respect to intended direction. WarpDriver agents are able to consistently head towards their intended direction. Other algorithms’ agents make very strong adaptations to their motions. Full graphs in Appendix B. Bottom: percentage of simulated frames containing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

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 produced many collisions between the agents and 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, as compared to less than 0.5% for ORCA and 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. As a result, WarpDriver agents are able to correctly anticipate and avoid collision with other agents, resulting in more natural reaction by agents (low deviation from intended directions) and none of them back-tracking.

7.4 Highly-Constrained Space

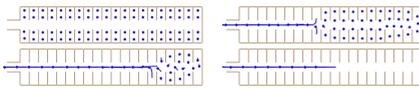


Figure 19: Plane example: plane exit situation involving 80 agents. Simulated with WarpDriver.

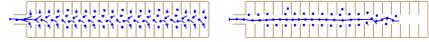


Figure 20: Issues encountered in Plane: Powerlaw agents from aisle seats exit before everyone else.

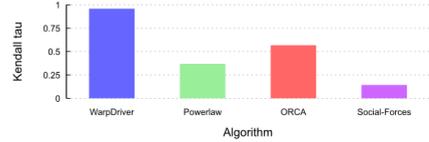


Figure 21: Kendall tau coefficient for plane exit order, higher is better. WarpDriver produces an order very close to the expected one, while other algorithms produce much further orders.

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 behavior leads to an orderly exiting process, where all agents are progressively evacuated, as evidenced by the high Kendall tau coefficient (0.96) which indicates the exit sequence is close to the ideal one (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 aisle 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.

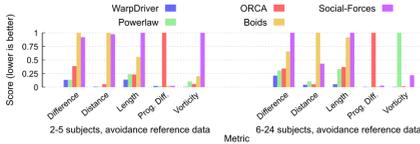


Figure 22: Benchmarks results using the method from [Wolinski et al. 2014], lower is better.

Data-driven validation Finally, we compared 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 basic 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.

In practice, assuming the typical target framerate of 15-20 fps 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 ~ 140 fps. Powerlaw on the other hand, for stability reasons requires much lower timesteps – values of ~ 0.005 sec can be found in the examples bundled with the source code, which means it needs to run at 200 fps or more to be real-time – and falls to ~ 40 fps on the *Big Groups* example that involves 2054 agents.

8 Discussion and Limitations

We present a novel probabilistic motion prediction algorithm for crowd simulation that accounts for the contextual interaction between the agent and its surroundings, including other agents, the environment layout, motion anticipation, etc.

We assume that the environments can be annotated with probable routes to be followed by agents. This step 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 Voronoi diagrams or any technique to compute static obstacles’ medial axes, or even learned from real data (e.g. camera feeds). More interestingly, our representation could be extended with route selection probabilities.

Although in this paper we have focused 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 concerning our probabilistic modeling is the risk of collision. The current implementation does not distinguish among various collision sources. As a result, for example, equivalent collision probabilities between a neighboring 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 obstacles. Extending the notion of risk of collision would allow us to mix into our simulations other types of moving obstacles (e.g. cars) with their corresponding level of danger.

Addressing each of these issues 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.

9 Conclusion

In this paper, we have presented a new context-aware motion prediction algorithm for crowd simulation. The main results of this approach are two-fold.

First, given its non-deterministic and probabilistic representation for motion prediction, agents do not perceive future collisions in a binary manner like in most of the existing methods; instead, they perceive a probability field of all future collisions. This model offers several advantages: (1) This characteristic results in smoother motion thanks of the continuity of the probability fields. Agents adapt their motion to lower the probabilities of colliding by following the gradient of the probability field. (2) Some agent’s oscillations between two binary future collision states often observed in some previous techniques are avoided. (3) Our anticipation considers several possible hypotheses, the notion of routes can be used when future position probabilities are propagated in time. (4) The non-determinism allows us to simulate uncertainty due to sensing or variety in locomotion trajectories. As we increase the uncertainty of agents’ future positions the further they are in time, we change the relative importance of agents that may collide sooner as opposed to those that may collide later.

The second innovation is related the contextual awareness of our technique, which not only depends on agents’ states, but also on

- external and contextual cues. This insight introduces a major difference from previous methods that assume agents keep moving with the same current velocity vector. One can easily conceive that agents' current velocity vectors are, most of the time, not representative of the intention of future motion, especially in crowds, where we are constantly adapting our locomotion trajectory to the presence of others.
- Through a set of challenging benchmark scenarios, as well as quantitative evaluations, we have demonstrated that this new probabilistic theoretic framework for motion prediction considerably improves the quality of visual simulations of crowds, and alleviates visual artifacts commonly observed in some state-of-the-art collision-avoidance algorithms.
- There are several avenues for future research. We would like to adapt our simulator to consider other forms of local interactions in addition to collision avoidance. One promising research direction is to learn future position likelihoods based on real observations. This would allow us 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.
- ## References
- CHENNEY, S. 2004. Flow tiles. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 233–242.
- CIVIDINI, J., APERT-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, ISA '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*, ACM, New York, NY, USA, SCA '09, 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, 6 (Nov.), 190:1–190:11.
- 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. 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, 140:1–140:10.
- 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, ISD '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, MIG '09, 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., CRÉTEUIL, 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., PETTRÉ, 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., CRÉTEUIL, 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, New York, NY, USA, SCA '09, 189–198.
- REYNOLDS, C. W. 1987. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Computer Graphics* 21, 4, 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, New York, 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*, 1928–1935.
- 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.