Velocity-Based Modeling of Physical Interactions in Multi-Agent Simulations

Sujeong Kim* University of North Carolina at Chapel Hill
Stephen J. Guy† University of Minnesota
Dinesh Manocha‡ University of North Carolina at Chapel Hill
http://gamma.cs.unc.edu/CrowdInteractions/

Abstract

We present an interactive algorithm to model physics-based interactions in multi-agent simulations. Our approach is capable of modeling both physical forces and interactions between agents and obstacles, while allowing the agents to anticipate and avoid collisions for local navigation. We combine velocity-based collision-avoidance algorithms with external physical forces. The overall formulation can approximately simulate various physical effects, including collisions, pushing, deceleration and resistive forces. We have integrated our approach with an open-source physics engine and use the resulting system to model plausible behaviors of and interactions among large numbers of agents in dense environments. Our algorithm can simulate a few thousand agents at interactive rates and can generate many emergent behaviors. The overall approach is useful for interactive applications that require plausible physical behavior, including games and virtual worlds.

CR Categories: I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—Multiagent systems

Keywords: Multi-Agent Simulation, Physical interactions

1 Introduction

Multi-agent simulations are frequently used to model a wide variety of physical systems, including human crowds; traffic; groups of birds, bees, fish, ants; and etc. In many of these applications it is important for the agents to interact in a physical manner with each other and the environment. Agents often collide, push, and impart forces on other agents and on the obstacles in the environment, changing their trajectory or behavior. A challenging goal is to model these interactions in large multi-agent systems at interactive rates. Many algorithms based on behavior modeling, social forces, cellular automata, and velocity-based formulation have been proposed for multi-agent simulation. Most of these techniques, however, focus on only the local navigation for each agent, and do not explicitly take into account physical interactions between agents or between agents and obstacles in the environment.

At a high level, there are two different sources of physical forces which may affect an agent's trajectories: interactions with other agents and interactions with objects in the environments. For example, dynamic objects such as falling boxes or moving cars may collide with an agent, pushing it from its path. Likewise, an agent may be pushed by, or bump into, other agents in dense scenarios. This can happen because of agent's intention (e.g. aggressive agent) or because the agent was pushed by an external force.

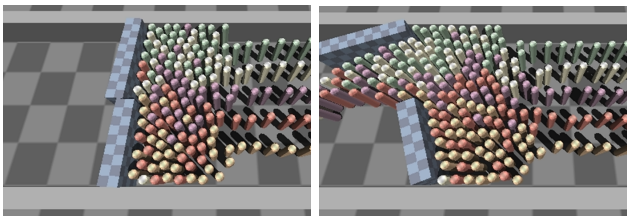


Figure 1: ***Wall Breaking****. We demonstrate the physical forces applied by cylindrical agents to breakable wall obstacles. Our algorithm can model such interactions between the agents and the obstacles in dense scenarios at interactive rates.*

While physical forces impact an agents trajectory, the agents motion will also impart forces upon the objects in his environment. This effect becomes increasingly important when the forces from many individual agents combine to produce a large effect on the environment, such as when dense, aggressive crowds bend fences or break walls. In order to simulate such scenarios, we need to develop appropriate two-way coupling techniques between autonomous agents and their physical environment.

Main Results: In this paper, we present a new method to model physical interactions between agents and objects in an interactive velocity-based multi-agent framework. Our approach incorporates both an agent's ability to anticipate the motion of other agents and avoid collisions using velocity obstacles and respond to physical forces in a single unified framework. We formulate the computation of velocity of each agent for each timestep as a linear programming problem in the velocity space. The linear constraints are computed by approximating the motion induced on an agent through Newtonian dynamics. This allows agents to account for forces from their environment and from other agents and generate a plausible trajectory. The resulting approach is efficient and can be used to simulate dense scenarios with thousands of agents at interactive rates. We have integrated our approach with the Bullet Physics Engine [AMD 2012], and reciprocal velocity obstacles [van den Berg et al. 2011], and demonstrate its performance in many complex scenarios with large number of agents and multiple moving obstacles, In practice, our approach can be used to generate physically plausible behavior for interactive multi-agent simulation.

The rest of the paper is organized as follows. Section 2 gives a brief review of related work. Section 3 gives an overview of our approach, which combines velocity-based multi-agent simulation and rigid body dynamics. We describe our approximate approach to computing velocity constraints using Newtonian dynamics for agent-agent interaction in Section 4 and for agent-obstacle interaction in Section 5. In Section 6, we highlight the performance on different scenarios and compare it with other techniques.

*e-mail:sujeong@cs.unc.edu
†e-mail:sjguy@cs.umn.edu
‡e-mail:dm@cs.unc.edu

2 Related Work

In this section, we give a brief overview of some related work in multi-agent and physically-based character simulation.

2.1 Multi-Agent and Crowd Simulation Models

Many approaches have been proposed to simulate the motion of large number of agents and crowds. Often these models are based on rules, which are used to guide the movement of each agent. An early example of such an approach is the seminal work of Reynolds [1999], which uses simple rules to model flocking behavior.

Force-based methods, such as the social force model [Helbing and Molnár 1995], use various forces to model attraction and repulsion between agents. These forces are not physically based; rather, they provide a mechanism to model the psychological factors that govern how agents approach each other. Helbing et al. [2000] model panic behavior with two additional physical forces (body force and sliding friction) in addition to the social forces. Yu and Johansson [2007] model the turbulence-like motion of a dense crowd by increasing the repulsive force. Other approaches model collision-avoidance behavior with velocity-based techniques [van den Berg et al. 2011; Pettré et al. 2009; Karamouzas and Overmars 2012] or vision-based steering approaches [Ondřej et al. 2010].

Other techniques have been proposed to model complex social interaction. HiDAC [Pelechano et al. 2007] uses various rules and social forces to model interactions between agents and obstacles; collision avoidance and physical interactions between agents and objects are handled using repulsive forces. The composite agent formulation [Yeh et al. 2008] uses geometric proxies to model social priority, authority, guidance, and aggression. Many other multi-agent simulation algorithms use techniques from sociology [Musse and Thalmann 1997], biomechanics [Guy et al. 2012], and psychology [Sakuma et al. 2005; Durupinar et al. 2011; Guy et al. 2011; Kim et al. 2012] to model different aspects of agent behaviors and decision models. These approaches are able to generate realistically heterogeneous behaviors for agents. Our approach to model physical interactions can also be combined with many of these approaches.

Many researchers have proposed cognitive and decision-making models to generate human-like behaviors [Shao and Terzopoulos 2005; Yu and Terzopoulos 2007; Ulicny and Thalmann 2002], or use data-driven approaches to the problem [Lee et al. 2007; Lerner et al. 2009].

Other approaches for modeling crowds are based on continuum or macroscopic models [Hughes 2003; Treuille et al. 2006; Narain et al. 2009]. In particular, Narain et al. [2009] present a hybrid technique using continuum and discrete method for aggregate behaviors in large and dense crowds. They are mainly used to simulate the macroscopic flow and do not model the interaction between the crowd and obstacles. In contrast, our approach simulates agent-agent and agent-obstacles physical interaction.

2.2 Force-Based Techniques for Character Animation

There has been extensive work on using physics-based models to improve character animation. Sok et al. [2010] use a force-based approach to ensure that the resulting motions are physically plausible. Other approaches consider geometric and kinematic constraints [Shum et al. 2012] or use interactive methods for character editing [Kim et al. 2009]. These techniques, which are primarily based on enhancing motion-captured data, can be used to simulate behaviors of and interactions between the characters and obstacles in their environment.

Many hybrid techniques have been proposed that bridge the gap between physics-based simulation of character motion and pre-recorded animation of characters to model responsive behavior of character [Shapiro et al. 2003; Zordan et al. 2005]. Muico et al. [2011] propose a composite method to improve the responsiveness of physically simulated characters to external disturbances by blending or transitioning multiple locomotion skills.

Our approach is quite different from these methods. Unlike character animation techniques that mainly focus on generating the full-body motion of a relatively small number of characters, we focus on generating physically plausible interactions between a large number of agents in dense scenarios.

2.3 Crowd Simulation in Game Engines

Some commercial game engines or middleware products can simulate character motion or crowd behavior. This includes Natural Motion's Euphoria, which simulates realistic character behavior based on biomechanics and physics simulation. There are also commercial AI middlewares for game engines that combine crowd and physics simulation: Kynapse, Havok AI, and Unreal Engine are examples of these. These systems primarily focus on the local and global navigation of each agent using navigation meshes and local rules. Our approach to generating physical interactions can be combined with these systems to improve local interactions between the agents and the obstacles in the scene.

3 Overview

Our framework simulates *agents* and *objects* differently, based on two fundamental assumptions about the nature of their motion. Agents are assumed to be autonomous and self-actuated. In the absence of external forces, we use velocity-based collision avoidance techniques to control the paths of the agents, who avoid collisions using anticipatory techniques. In contrast, objects in the environment move only when physical forces act on them. The positions of objects are updated by solving Newton's equations of motion; contacts are handled with a constraint-based method. This section gives an overview of our proposed approach to simulating agents and objects together in a shared space.

Local navigation and anticipatory collision avoidance of agents can be efficiently modeled using reciprocal velocity obstacles, which imposes linear constraints on an agent's velocity to help it navigate its environment. We extend this framework by representing the effect of physical forces on agents also as linearized velocity constraints. This allows us to use linear programming to compute a new velocity for each agent – one which takes into account both the navigation and force constraints imposed upon that agent.

Agent simulation is typically performed over discrete timesteps. Agents are assumed to have a *preferred velocity*. This is the velocity at which the agent would travel if there were no anticipatory collision-avoidance or physical constraints. This velocity is used to define the cost function for linear programming or constrained optimization. At each timestep, an agent computes a new velocity that satisfies the velocity constraints, then updates its position based on this velocity. A new set of velocity constraints are then computed based on the new positions and velocities.

There are two types of constraints which we impose on an agent's velocity:

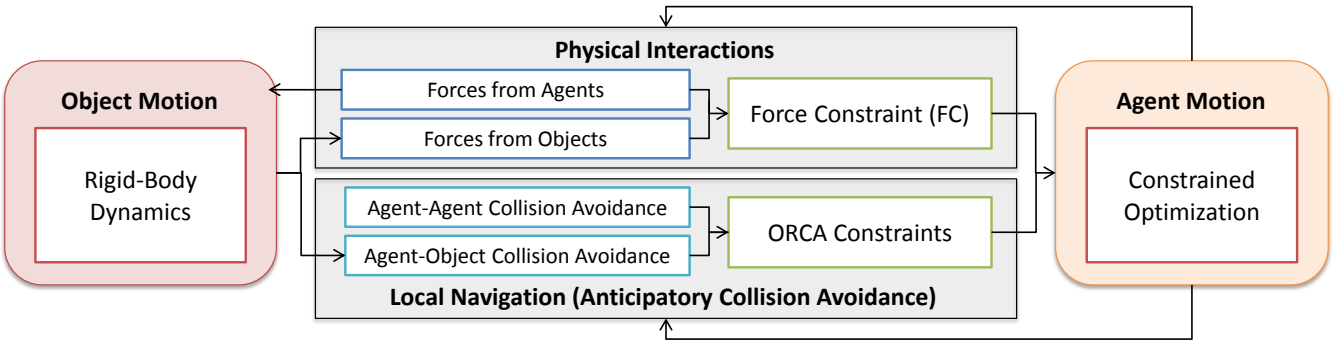


Figure 2: ***System Overview.*** *The motions for objects and agents are computed by a rigid-body dynamics solver and a constrained optimizer, respectively. Physical interactions between agents and obstacles determine forces. For obstacles, the forces serve as inputs to the rigid-body system; for agents, they become force constraints. These force constraints are combined with the original ORCA planning constraints and serve as inputs to optimization algorithm.*

- **ORCA Constraints** define the space of velocities which are expected to remain collision-free for a given period of time. The derivation of agent-agent ORCA constraints is given in Section 3.1, and that of agent-object ORCA constraints in Section 5.1.1.
- **Force Constraints** are constraints which arise out of forces initiated by physical interactions with other agents and objects. The details are given in Sections 3.2, 4, and 5.

Fig. 2 gives an overview of the full simulation system for computing these constraints and for updating an agent's position and velocity.

3.1 Velocity Constraints for Local Navigation

ORCA constraints are defined by a set of velocities that are guaranteed to avoid upcoming collisions with other nearby agents [van den Berg et al. 2011]. The constraints are represented as the boundary of a half plane containing the space of feasible, collision-free velocities. Given two agents, A and B, which we represent as 2D discs, we compute the minimum vector $\mathbf{u}$ of the change in relative velocity needed to avoid collision. ORCA enforces this constraint by requiring each agent to change their current velocity by at least 1/2 $\mathbf{u}$. The ORCA constraint on A's velocity induced by B would be:

$$ORCA_{A|B} = \{\mathbf{v} | (\mathbf{v} - (\mathbf{v}_A + \frac{1}{2}\mathbf{u})) \cdot \hat{\mathbf{u}} \geq 0\}, \qquad (1)$$

where $\mathbf{v}_A$ is A's current velocity and $\hat{\mathbf{u}}$ is the normalized vector $\mathbf{u}$.

If A has multiple neighboring agents, each will impose its own ORCA constraint on A's velocity. Local navigation is computed by finding the new velocity for A ($\mathbf{v}^{new}$) which is closest to its preferred velocity ($\mathbf{v}^{pref}$) while respecting all the ORCA constraints.

3.2 Velocity Constraints from Physical Forces

The set of neighbors involved in physical interactions with an agent include both nearby agents and obstacles. We define a radius and an angle that are then used to define a range of physical interactions for each agent. When an agent is pushed, either by another agent or by an obstacle, the agent experiences an external force. By Newton's second law, the net force acting on an agent implies a net acceleration. Given a known timestep, we can compute the change in velocity exactly. We represent this change in velocity induced by a force as an additional constraint on the agent velocity.

One benefits of applying forces as a form of constraint is the ability of an agent to adapt to the forces. While the constraint guarantees an acceleration at least as large as that implied by the dynamics, the actual acceleration from the forces may be greater than that. When pushed, rather than simply falling sideways, an agent could accelerate faster to reach a stable, controlled state.

We classify these forces into two types, depending on the origin of the force:

Forces from agents are generated when an agent pushes (or is pushed into) another agent, or when there is a collision between two agents. This effect of a pushing force can persist across multiple time-steps depending on the agent's response. The effect of a pushing force on an agent can also propagate to other agents as a result of the first agent's being pushed into others. The force imparted by the agent onto an object is given as an input to a rigid body dynamic simulation, which we use to simulate the behavior of the objects in the environment. This simulation accounts for the impact of agent's force on the motion of the object.

Forces from objects are the forces an agent receives from objects. Note that forces acting upon agents from objects are only those caused by the collision, i.e. the reaction force. These forces are then summed up and represented as a Force Constraint, an additional constraint to the velocity computation.

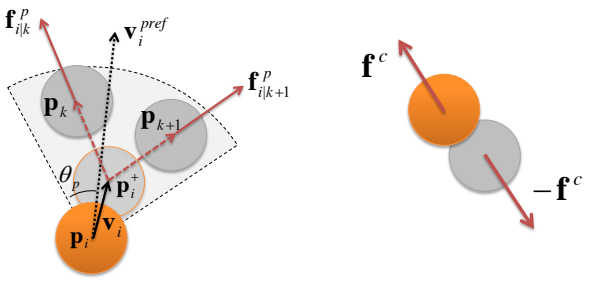
3.3 Velocity Computation Algorithm

We can summarize our new velocity computation algorithm as follows: Given an agent A with neighbors B, we define the *permitted velocities* for A, PV_A as the intersection of all velocity constraints. We can state our agent update algorithm as an optimization problem. Formally:

$$PV_A = FC_A \cap \bigcap_{B \neq A} ORCA_{A|B} \qquad (2)$$

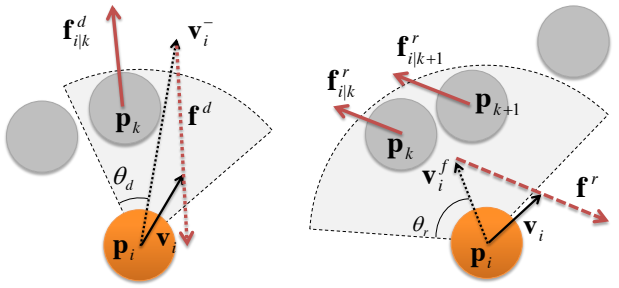
$$\mathbf{v}^{new} = \underset{\mathbf{v} \in PV_A}{\operatorname{argmin}} \|\mathbf{v} - \mathbf{v}^{pref}\|. \qquad (3)$$

In conditions where the preferred velocity of an agent is only determined by physical forces (i.e., in the absence of navigation constraints) the formulation will reproduce the motion based on Newton's second Law. This is because the closest velocity to the agent's current velocity will be the perpendicular distance to the velocity constraint FC_A.



(a) *Pushing Force* **(b)** *Collision Force*

Figure 3: ***Contact Forces*** *An agent (orange disk) can affect nearby neighbors (grey disks) through physical contact expressed as impulse forces. These physical forces can be used to push other agents as in (a) or resolve physical collisions as in (b). The red arrows display the direction of the resulting forces.*



(a) *Deceleration Force* **(b)** *Resistive Force*

Figure 4: ***Inferred Forces****: Forces between agents can be inferred based on local navigation. If an agent has a large change in velocity in the absence of force applied to the agent, as in (a), then a deceleration force, $\mathbf{f}^d$, is inferred to have caused the change, and is applied to nearby agents. (b) Likewise, when a force is applied to an agent which produces no change in agent's velocity, we model in terms of resistive force $\mathbf{f}^r$, which implicitly opposes this motion. This inferred resistive force is also applied to nearby agents.*

4 Force Computation

In this section, we present our approach for computing velocity constraints from physical forces. We propose an approximate approach because we need to handle a large number of agents in dense environments. As a result, we approximate the physical interactions based on appropriate velocity constraints. We assume that these forces are initiated from a collision or by pushing. When an agent experiences these forces, the impact on its motion lasts more than one timestep because of its effort to recover momentum. We approximate the effects of momentum by using two *inferred forces*: a resistive force and a deceleration force. These two additional forces have the net effect of propagating the momentum through the crowd.

4.1 Contact Forces

Pushing Forces: Pushing is one of the ways for agents to *physically* interact with each other [Pelechano et al. 2007]. Agents can impart a pushing force on nearby agents. In our formulation, agents can impart a pushing force to nearby agents; this pushing force follows the approximate direction of the pushing agent's preferred velocity and pushes the blocking agent out of the pushing agent's path (see Fig. 3a). The pushing force imparts an impulse to the nearby agents in the direction of the normal vector from the pushing agent towards the pushed agent. Formally, the pushing force $\mathbf{f}^p_{i|k}$ exerted by an agent i pushing another agent k can be given as:

$$\mathbf{f}^p_{i|k} = \rho_k f_p \frac{\mathbf{p}_k - \mathbf{p}_i^+}{\|\mathbf{p}_k - \mathbf{p}_i^+\|}, \tag{4}$$

where $\mathbf{p}_i$ and $\mathbf{p}_k$ indicate the positions of agent i and k, respectively, and $\mathbf{p}_i^+ = \mathbf{p}_i + \mathbf{v}_i \Delta t$ is the pushing agent's future position at the next time step. f_p is used to define the weight of pushing force, and we formulate it as an inverse of number of agents that are pushed.

Collisions: In case of collisions between agents, a collision resolution force is applied (Fig. 3b). This force is computed based on the physically-based simulation approach proposed by [Baraff 1997], and depends on the mass and velocity of colliding agents. We consider only linear momentum and simulate agents as radially symmetric disks. As a result, we do not take into account the orientation of the agents. For an agent i colliding with agent k, the collision force $\mathbf{f}^c$ is computed as follows:

$$\mathbf{f}^c = \left(\frac{-(1+\epsilon)\mathbf{v}^{rel}}{1/m_i + 1/m_k} \cdot \mathbf{n}\right)/\Delta t, \tag{5}$$

where $\mathbf{n}$ is the collision normal, pointing towards agent i from agent j; v^{rel} is relative velocity; and m_i and m_k are the mass of agent i and agent k, respectively. ϵ is the coefficient of restitution.

4.2 Inferred Forces

We also define two forces, deceleration force and resistive force, to model agent's ability to adjust their motion when external force is applied. The contact forces that result from agents colliding or pushing each other are computed as impulses. After being bumped or pushed, an agent will naturally exert forces in order to quickly recover its preferred velocity. Forces will therefore propagate through a dense crowd, since one agent is likely to push others in order to recover from the external pushing force.

This kind of behaviors are inspired from biomechanics, an observation about how humans react to recover their balance in various conditions including when the external forces are applied to the body. More details are given in [Kim et al. 2013].

These propagation forces can be inferred when the motion computed using constrained optimization does not match the motion expected from external physical forces. For example, when an agent decelerates at a faster rate than that implied by the external forces, we infer that the agent must be pushing against other agents or obstacles in order to be able to slow down so quickly. Likewise, when an agent accelerates at a rate less than that implied by external forces, we infer the agent must be pushing against other agents or obstacles, which resist the effect of the forces. These inferred propagation forces are applied to the appropriate neighboring agents during the subsequent timestep. We describe the formulation for each of these forces below.

Deceleration Forces: When an agent reduces speed while preserving direction to within a certain threshold (θ_d), we introduce a force into the system based on this velocity change. The deceleration force generated by agent i's deceleration is defined as:

$$\mathbf{f}^d_i = \begin{cases} k_{thresh} m_i \Delta \mathbf{v}_i / \Delta t & \text{if } (\Delta \hat{\mathbf{v}}_i \cdot \hat{\mathbf{v}}_i) < -cos(\theta_d), \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where $\Delta \mathbf{v}_i = \mathbf{v}_i - \mathbf{v}_i^-$ is the change in velocity from the previous time step to the current time step. Because agents are not rigid bodies, they can absorb or transform forces. We approximate this behavior by introducing a parameter k_{thresh}.

We assume that the speed reduction arises from one of two sources: self-will (e.g. sudden change of preferred velocity) or agent interaction (e.g. impending collision avoidance). When there are no interacting agents, we assume it is the former case, and the deceleration force is applied back to the agent itself. In the latter case, where the deceleration is caused by interaction with the agents neighbors, the behavior of those neighbors should also change as a result of the interactions; we thus distribute the deceleration force among them in the case of collision avoidance. Furthermore, a neighboring agent k causes such behavior if it lies within a cone centered on $\mathbf{v}_i^-$ and is within an angular space of $2\theta_d$ degrees (as shown in Fig. 4a). For each interacting neighbor k of agent i, the portion of the deceleration force acting on agent k is defined as:

$$\mathbf{f}_{i|k}^d = -\delta_k \mathbf{f}_i^d, \tag{7}$$

where δ_k is a parameter that indicates how the deceleration force is transferred to agent k. We set this parameter to $1/n$, where n is the number of interacting agents.

Resistive Forces: Resistive forces occur when an agent's computed velocity does not account for the entire change in velocity expected from the external force. This difference is propagated to neighboring agents via the resistive forces. This force is computed by the difference between the velocity $\mathbf{v}$ computed by (3) and the velocity $\mathbf{v}^f$ computed only from the net force applied to the agent. The resistive force of an agent i experiencing the discrepancy between $\mathbf{v}^f$ and $\mathbf{v}$ is:

$$\mathbf{f}_i^r = \begin{cases} k_{thresh} m_i (\mathbf{v}_i - \mathbf{v}_i^f)/\Delta t & \text{if } \mathbf{v}_i^f \neq \mathbf{0} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

As in the case of deceleration force, the resistive force is applied to the agent i when there is no interacting agent. Otherwise, the resistive force is distributed equally among the interacting agents, whose position is inside a cone centered on $\mathbf{v}_i^f$ and with an angular span of $2\theta_r$ degrees (as shown in Fig. 4b). The resistive force $\mathbf{f}_{i|k}^r$ applied to agent k is given as:

$$\mathbf{f}_{i|k}^r = -\gamma_k \mathbf{f}_i^r, \tag{9}$$

where γ_k is a weighting parameter for agent k (we use $1/n$).

The resistive force and deceleration force can be viewed as complementary to one another. The resistive force is non-zero only in the presence of external physical forces on an agent, and the deceleration force is non-zero only in the absence of such forces.

4.3 Force Constraints

The net force $\mathbf{f}$ is the sum of all the forces applied to the agent. Mathematically, force $\mathbf{f}$ used to compute force constraint FC (described in (12)) is computed as follows:

$$\mathbf{f} = \sum \mathbf{f}^c + \sum \mathbf{f}^d + \sum \mathbf{f}^r + \sum \mathbf{f}^p. \tag{10}$$

The force constraint FC induced by the net force $\mathbf{f}$ is computed as follows:

$$\mathbf{v}^f = \mathbf{v} + \frac{\mathbf{f}}{m}\Delta t \tag{11}$$

$$FC = \{\mathbf{v} | (\mathbf{v} - \mathbf{v}^f) \cdot \hat{\mathbf{f}} \geq 0\}. \tag{12}$$

FC is a half plane whose boundary, a line through $\mathbf{v}^f$, is perpendicular to the normalized force $\hat{\mathbf{f}}$. This half plane contains a set of velocities that is equal to or greater than the minimum velocity change required by the force $\mathbf{f}$. This term is used for velocity computation in Equation (2).

5 Interaction with Obstacles

A key part of our approach is to model interactions between the agents and static and dynamic objects, i.e. two-way coupling between agents and obstacles. The behavior of agents towards the objects around them includes anticipatory collision avoidance, pushing, and unintended collisions. An agent might also impose forces from its motion (e.g., resistive force and deceleration force) on obstacles, as it does to other agents. If there is a collision, then objects also exert forces on the agent. In this section, we present an efficient algorithm to model these interactions for interactive applications.

5.1 Dynamic Objects

There are some significant differences between agent-agent and agent-obstacle interactions, both in terms of the motion computation and in how an agent responds to those obstacles. Importantly, the motion of obstacles (e.g. rigid bodies) is governed by Newtonian physics, since these objects have no will and are unable to initiate movement on their own. As a result, the agents cannot assume that the obstacles will anticipate collisions and change trajectory to avoid them. Moreover, the rigid body simulation is performed on the obstacle motion in 3D space, while the agents are constrained to move on a 2D plane.

5.1.1 Anticipatory Collision Avoidance

In our approach, agents attempt to anticipate and avoid collisions with the obstacles. Since the agent's navigation is performed in 2D space, we project the boundary of the dynamic obstacle onto the 2D plane (see Fig. 5).

The dynamic object O is represented, like the agents, as an open disc centered at $\mathbf{p}$ with the radius r of the bounding sphere of the object. While we use this bounding shape for collision avoidance with the agents, the underlying rigid body simulation uses an exact 3D object representation for collision detection and for response to other objects in the scene.

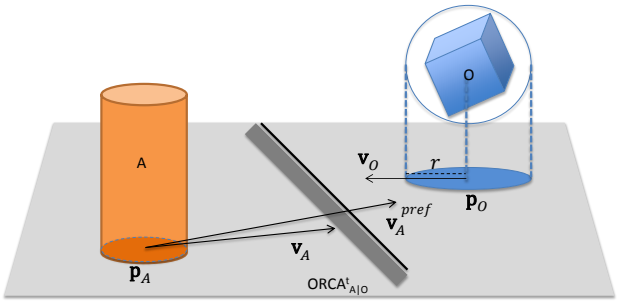


Figure 5: ***Collision Avoidance and Anticipation with a 3D object projected onto 2D plane*** *We take into account the object location in computing appropriate collision avoidance constraints for agent A, shown in the shaded region.*

Agents try to avoid collisions with dynamic obstacles, just as they try to avoid collisions with other agents, whenever the dynamic obstacles are within agent's visual range. However, agents do not

assume objects will reciprocate in avoiding collisions. Therefore, assuming that a change in velocity of $\mathbf{u}$ (Section 3.1) is required to avoid an anticipated collision with an obstacle, the agent will modify its velocity by at least $\mathbf{u}$; this is twice as large as the velocity bound using ORCA algorithm.

Therefore, the collision avoidance constraint for agent A induced by object O is:

$$ORCA^{\tau}_{A|O} = \{\mathbf{v}|(\mathbf{v} - (\mathbf{v}_A + \mathbf{u})) \cdot n \geq 0\}. \quad (13)$$

5.1.2 Agent-Object Collisions

When there is a collision between an agent and an object, the impulse force $\mathbf{f}^c$ is computed by the method used in [Baraff 1997]. We only consider rotational factors in the computation of object motion, not for the agents. We can compute the impulse force $\mathbf{f}^c$ from the collision between an agent a and object o is as follows:

$$\mathbf{f}^c = (\frac{-(1+\epsilon)\mathbf{v}^{rel}}{1/m_a + 1/m_o} \cdot \mathbf{n})/\Delta t, \quad (14)$$

where m_o is the mass of object o, $\mathbf{v}^{rel}$ and $\mathbf{n}$ are the relative velocity and the contact normal between the contact points, respectively. A force with the same magnitude but with the opposite direction is applied to the object, which also results in change of angular motion generated by the torque τ^c:

$$\tau^c = \mathbf{f}^c \times \mathbf{r}_o, \quad (15)$$

where $\mathbf{r}_o$ is the displacement vector for the contact point of the object.

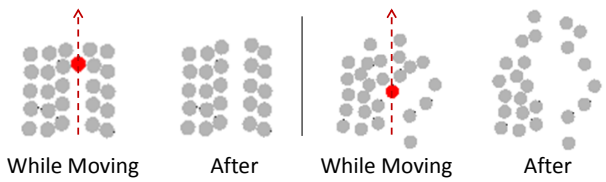
6 Results & Analysis

In this section, we highlight the performance of our algorithm in different scenarios. We also analyze the approach and compare it with other techniques.

6.1 Agent-Agent Interaction

We first demonstrate a few scenarios which highlight the effect of forces propagating in agent-agent interactions.

Running Through Scenario: We demonstrate a scenario where an agent runs at a high speed through a dense crowd of 25 agents that are standing still. Figure 6 compares the result of our method to those achieved using multi-agent simulation without any physical interactions.



(a) *without physical interactions* **(b)** *with physical interactions*

Figure 6: ***Rushing through still agents****: The red agent tries to rush through a group of standing agents, simulated (a) with only anticipatory collision avoidance and (b) with physical interactions. Using our method, the forces are propagated among the agents, resulting in a new distribution pattern (b).*

The left side of each image shows a pushing agent (red) passing through the crowd, and the right side of each image shows the position of all other agents in the crowd after the fast-moving agent has passed. As Fig. 6 demonstrates, agents simulated without physics-based interaction use minimal motion to avoid collisions. In contrast, agents simulated using our physically-based formulation resist the pushing motion (in an attempt to stand still) and propagate the effects of being pushed to other agents.

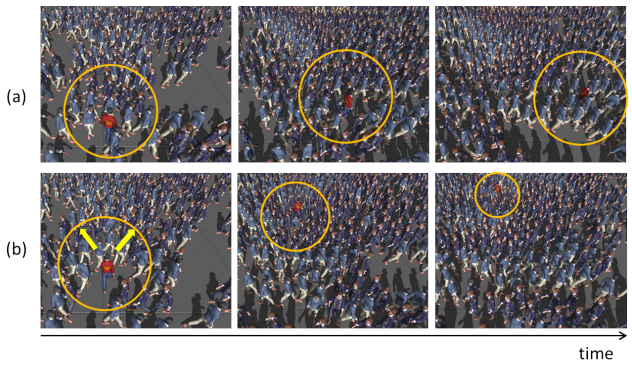


Figure 7: ***Pushing through dense crowd****: The red agent pushes through a dense crowd that moves perpendicular to its direction of travel. Agents are simulated using (a) ancipatory collision avoidance only, and (b) combination of anticipatory collision avoidance and physically-based interaction. In the latter case, the red agent can proceed to its goals quickly by pushing other agents through its path.*

Dense Crossing Scenario: In this case, an agent attempts to cross perpendicularly through a dense stream of crowd flow. Fig. 7 shows a comparison between our method and using no physical interactions.

As the figure shows, an agent who is only avoiding collisions (without pushing) cannot effectively cut through the crowd's flow, is eventually swept up with the crowd, as that motion avoids all impending collision. This is because moving with the crowd successfully avoids all impending collisions. However, the pushing force based on our approach allows an agent to clear its path and move freely.

Two Bottlenecks Scenario: In this scenario, long lines of closely spaced walking agents attempt to pass through two narrow bottlenecks, as illustrated in Fig. 8. The first bottleneck (shown as (2)) is about the width of two agents; the second is narrower, about wide enough for one agent (shown as (1)). A local navigation algorithm that performs collision avoidance frequently results in congestion at both the bottlenecks due to stable-arch formation of agents (highlighted with a yellow circle) in Fig. 8 (a). However, agents simulated by our physically-based method are able to break this congestion at the bottleneck area by pushing the blocking agents. The ability to break through bottlenecks also results in a quantitatively higher rate of flow for agents using our approach. After seconds, twice as many agents make it through both the bottlenecks, using our algorithm.

6.2 Agent-object Interaction

We can also demonstrate the effect of two-way coupling between dynamic objects and agents in multi-agent simulations. In the following scenarios, the Bullet Physics engine [AMD 2012] is used to compute the 3D rigid body dynamics, which in turn are used to

(a) *Multi-agent simulation with no physical interaction*

(b) *Physical interaction amongst agents and with the walls*

Figure 8: ***Two bottlenecks scenario*** *We simulate and compare crowd behavior at two narrow bottlenecks in these scenarios, (1) and (2), which are marked with red dotted lines. Bottleneck (1) is barely wide enough for one person to pass through; bottleneck (2) is about twice that width and allows two agents to pass through it at a time. The result from collision-avoidance-only simulation results in an arch-shaped arrangement of agents in the crowd (highlighted with a yellow circle), which causes congestion at the bottleneck. Our method breaks the congestion by allowing the agents to push one other in congested conditions.*

compute object motion (see Fig. 2). The effects of user interaction in these scenarios can be seen in the supplemental video[1].

Rolling Ball Scenario: In this scenario, a few agents interact with varying numbers of dynamically generated balls. A user can interact with the agents by moving around the dynamic obstacles, or by generating new balls. Agents attempt to avoid these dynamically moving balls and push them away when there is a collision.

Wall Breaking Scenario: In this scenario, long lines of agents come at a constant rate into the simulated region, which is blocked off with a movable wall made of 200 blocks glued together. This wall can be broken into separate blocks if a large external force is applied by the agents. Agents initially stop to avoid hitting the wall, but as other agents start to push from behind, the wall breaks apart and gets carried away with the agents. Fig. 1 shows stills from the simulation.

Cluttered Office Scenario: In this scenario, several decomposed 3d models - a table, a chair, and a shelf, and several rigid bodies (e.g. boxes) stacked on top of each other – are placed in the way of the agents. A long stream of agents attempts to navigate past the obstacles. Users can throw boxes, which push the agents and knock over objects in the environment. Fig. 9 shows a still from the simulation.

These scenarios demonstrate several features of our approach:

- *Dynamic Obstacle Avoidance:* Agents try to avoid collisions with other agents and with dynamic obstacles.
- *Agent-Object Interactions:* Our method takes into account the collisions which occur between the agents and the objects. The forces generated by these collisions affect both the objects and the agents.
- *User Interactions:* Our method is fast enough for real-time interactive simulation. Users can participate in the simulation by moving rigid bodies inside the scene; this movement dynamically changes the environment for the moving agent.

[1]Supplementary video can be found at
`http://gamma.cs.unc.edu/CrowdInteractions/`

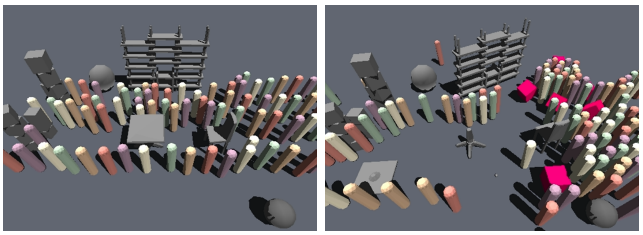


Figure 9: ***Office Scenario****. Agents navigate to avoid office furniture. As users insert flying pink boxes into the scene, the agents get pushed, collide into each other, and avoid falling objects (see video).*

6.3 Performance

We measured the simulation timings for the demos we presented (see Table 1). The timings were computed on a 3.4 GHz Intel i7 processor with 8GB RAM. Our method efficiently simulates large numbers of agents, and also exhibits interactive performance when integrated with the Bullet Physics Library.

Scenario	# Agents	# Dynamic Obstacles	# Static Obstacles	fps
Pushing Through	1600	0	0	229.6
Two Bottlenecks	1000	0	20	829.7
Rolling Balls	10	1000	2	1205.9
Wall Breaking	1200	200	2	50.1
Office	1200	65	0	69.0

Table 1: *Performance on a single core for different scenarios. Our algorithm can handle all of them at interactive rates.*

6.4 Analysis

Our approach is mainly designed for interactive applications that require plausible physical behavior (e.g. games or virtual worlds). By using a combination of force and navigation constraints that affect agents' behavior, our approach can simulate many use effects and emergent behaviors. For example, our formulation allows for intentionally uncooperative agents to physically push their way through a crowd by imparting physical forces to nearby agents. Additionally, agents can use navigation constraints to avoid collisions with dynamic obstacles as well as other agents. By expressing all interactions as linear velocity constraints, we can naturally combine the two different simulation paradigms of forces and navigation into a unified framework and compute the new velocity for each agent using linear programming. This is useful in generating physically plausible simulations of large numbers of agents.

Benefits of Our Method

Many techniques have been proposed in the literature for simulating large numbers of agents that display a wide variety of emergent behaviors. However, the primary emphasis of these methods is on collision avoidance – avoiding any physical contact between the

agents. In other words, they model how agents move around each other, but do not usually model explicit physical contacts, interactions, and external forces.

Force-based methods such as [Helbing and Molnár 1995] use forces to describe social factors (e.g. attraction and repulsion) between the agents, not physical interactions. Most closely related to our work are methods such as [Helbing et al. 2000; Yu and Johansson 2007; Pelechano et al. 2007], which model crowd turbulence or physical interactions among panicking agents by adding explicit physical force or by increasing repulsive forces. These methods are capable of reproducing some important emergent crowd phenomena, but do not account for the anticipation needed to efficiently avoid upcoming collisions with other agents and obstacles [Curtis et al. 2012].

Force-based methods can also suffer from stability issues in dense scenarios, which require careful tuning and small time steps in order to remain stable [Curtis et al. 2011]. Our method provides stable, anticipatory motion for agents while incorporating agent responses to forces. It can be easily combined with other velocity-based approaches. Our approach is also stable in terms of varying the size of time-steps. More details are given in [Kim et al. 2013].

Limitations

We use a physically-inspired approach to simulate the interactions between a high number of agents and the obstacles. However, it is only an approximation and may not be physically accurate. Secondly, we assume that agents are constrained to move along a 2D plane, and we use the projected positions of 3D dynamic objects to compute the interactions. Third, like other agent-based simulation methods, we use a rather simple approximation for each agent (a 2D circle). This means that we cannot accurately simulate physical interactions with human-like articulated models and 3D objects.

7 Conclusion and Future Work

We have proposed a novel method to combine physics-based interactions with anticipatory collision-avoidance techniques that use velocity-based formulation. Our method can generate many emergent behaviors, physically-based collision responses, and propagation of forces to the agent's nearby neighbors. In combination with the Bullet Physics library, we were able to simulate complex interactions between agents and dynamic obstacles in the environment. The resulting approach is useful for interactive large-scaled simulations and can generate physically plausible behaviors. This approach has been extended to model physical interactions between dense crowds and applied to Tawaf simulation [Kim et al. 2013].

In our future work, we would like to further explore our method by comparing the results with real-world crowd behaviors and by performing more validation. We would like also to extend our model to agents moving in 3D space or multi-layer frameworks, and to consider using more complex shapes, or even articulated body models, to represent agents, as this would allow for more accurate force computation. Finally, we would like to use more accurate physically-based modeling algorithms to generate appropriate behaviors.

Acknowledgements

We are grateful to the reviewers for their comments, we would like to thank Sean Curtis, Ming C. Lin and Ioannis Karamouzas for their help and feedback, and Karl Hillesland, Erwin Coumans, and Jason Yang from AMD for their support. This research is supported in part by ARO Contracts W911NF-10-1-0506, W911NF-12-1-0430, NSF awards 0917040, 0904990, 100057, and 1117127, AMD, and Intel.

References

AMD, 2012. Bullet Physics 2.80. http://bulletphysics.org.

BARAFF, D. 1997. An introduction to physically based modeling: Rigid body simulation i - unconstrained rigid body dynamics. In *In An Introduction to Physically Based Modelling, SIGGRAPH '97 Course Notes*, 97.

CURTIS, S., GUY, S. J., ZAFAR, B., AND MANOCHA, D. 2011. Virtual tawaf: A case study in simulating the behavior of dense, heterogeneous crowds. In *1st IEEE Workshop on Modeling, Simulation and Visual Analysis of Large Crowds*, 128–135.

CURTIS, S., ZAFAR, B., GUTUB, A., AND MANOCHA, D. 2012. Right of way. *The Visual Computer*, 1–16.

DURUPINAR, F., PELECHANO, N., ALLBECK, J., GÜ ANDDÜ ANDKBAY, U., AND BADLER, N. 2011. How the ocean personality model affects the perception of crowds. *Computer Graphics and Applications, IEEE 31*, 3 (may-june), 22 –31.

GUY, S. J., KIM, S., LIN, M. C., AND MANOCHA, D. 2011. Simulating heterogeneous crowd behaviors using personality trait theory. In *Symposium on Computer Animation*, ACM, 43–52.

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.

HELBING, D., AND MOLNÁR, P. 1995. Social force model for pedestrian dynamics. *Phys. Rev. E 51* (May), 4282–4286.

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

HUGHES, R. L. 2003. The flow of human crowds. *Annual Review of Fluid Mechanics 35*, 1, 169–182.

KARAMOUZAS, I., AND OVERMARS, M. 2012. Simulating and evaluating the local behavior of small pedestrian groups. *IEEE Trans. on Visualization and Computer Graphics 18*, 3, 394–406.

KIM, M., HYUN, K., KIM, J., AND LEE, J. 2009. Synchronized multi-character motion editing. *ACM Trans. Graph. 28*, 3 (July), 79:1–79:9.

KIM, S., GUY, S. J., MANOCHA, D., AND LIN, M. C. 2012. Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory. In *Symposium on Interactive 3D Graphics*, ACM, New York, NY, USA, I3D '12, 55–62.

KIM, S., GUY, S. J., ZAFAR, B., GUTUB, A., AND MANOCHA, D. 2013. Velocity-based modeling of physical interactions in multi-agent simulations in dense crowd. Tech. rep., Department of Computer Science, University of North Carolina at Chapel Hill.

LEE, K. H., CHOI, M. G., HONG, Q., AND LEE, J. 2007. Group behavior from video: a data-driven approach to crowd simulation. In *Symposium on Computer Animation*, 109–118.

LERNER, A., CHRYSANTHOU, Y., SHAMIR, A., AND COHEN-OR, D. 2009. Data driven evaluation of crowds. In *MIG*, 75–83.

MUICO, U., POPOVIĆ, J., AND POPOVIĆ, Z. 2011. Composite control of physically simulated characters. *ACM Transactions on Graphics 30*, 3.

MUSSE, S. R., AND THALMANN, D. 1997. A model of human crowd behavior: Group inter-relationship and collision detection

analysis. In *Proc. Workshop of Computer Animation and Simulation of Eurographics'97*, 39–51.

NARAIN, R., GOLAS, A., CURTIS, S., AND LIN, M. C. 2009. Aggregate dynamics for dense crowd simulation. *ACM Trans. Graph. 28*, 5 (Dec.), 122:1–122:8.

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.

PELECHANO, N., ALLBECK, J. M., AND BADLER, N. I. 2007. Controlling individual agents in high-density crowd simulation. In *Symposium on Computer animation*, 99–108.

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 *Symposium on Computer Animation*, ACM, SCA '09, 189–198.

REYNOLDS, C. 1999. Steering Behaviors for Autonomous Characters. In *Game Developers Conference 1999*.

SAKUMA, T., MUKAI, T., AND KURIYAMA, S. 2005. Psychological model for animating crowded pedestrians: Virtual humans and social agents. *Comput. Animat. Virtual Worlds 16*, 343–351.

SHAO, W., AND TERZOPOULOS, D. 2005. Autonomous pedestrians. In *Symposium on Computer animation*, 19–28.

SHAPIRO, A., PIGHIN, F., AND FALOUTSOS, P. 2003. Hybrid control for interactive character animation. In *Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, USA, PG '03, 455–.

SHUM, H. P. H., KOMURA, T., AND YAMAZAKI, S. 2012. Simulating multiple character interactions with collaborative and adversarial goals. *IEEE Transactions on Visualization and Computer Graphics 18*, 5 (May), 741–752.

SOK, K. W., YAMANE, K., LEE, J., AND HODGINS, J. 2010. Editing dynamic human motions via momentum and force. In *Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '10, 11–20.

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

ULICNY, B., AND THALMANN, D. 2002. Towards interactive real-time crowd behavior simulation. In *Computer Graphics Forum*, vol. 21, Wiley Online Library, 767–775.

VAN DEN BERG, J., GUY, S. J., LIN, M., AND MANOCHA, D. 2011. Reciprocal n-body collision avoidance. In *Robotics Research: 14th ISRR (STAR)*, vol. 70, 3–19.

YEH, H., CURTIS, S., PATIL, S., VAN DEN BERG, J., MANOCHA, D., AND LIN, M. 2008. Composite agents. In *Symposium on Computer Animation*, 39–47.

YU, W., AND JOHANSSON, A. 2007. Modeling crowd turbulence by many-particle simulations. *Phys. Rev. E 76* (Oct), 046105.

YU, Q., AND TERZOPOULOS, D. 2007. A decision network framework for the behavioral animation of virtual humans. In *Symposium on Computer animation*, 119–128.

ZORDAN, V. B., MAJKOWSKA, A., CHIU, B., AND FAST, M. 2005. Dynamic response for motion capture animation. *ACM Trans. Graph. 24*, 3 (July), 697–701.