The Visual Computer manuscript No.
(will be inserted by the editor)

Velocity-Based Modeling of Physical Interactions in Dense Crowds

Supplementary video: `http://youtu.be/KAnmNg2_hI4`

Sujeong Kim · Stephen J. Guy · Karl Hillesland · Basim Zafar · Adnan Gutub · Dinesh Manocha

Received: date / Accepted: date

Abstract We present an interactive algorithm to model physics-based interactions in dense crowds. Our approach is capable of modeling both physical forces and interactions between agents and obstacles, while also allowing the agents to anticipate and avoid upcoming collisions during local navigation. We combine velocity-based collision-avoidance algorithms with external physical forces. The overall formulation produces various effects of forces acting on agents and crowds, including balance recovery motion and force propagation through the crowd. We further extend our method to model more complex behaviors involving social and cultural rules. We use finite state machines to specify a series of behaviors and demonstrate our approach on many complex scenarios. Our algorithm can simulate a few thousand agents at interactive rates and can generate many emergent behaviors.

Keywords Multi-agent Simulation · Physical Interactions

S. Kim and D. Manocha
University of North Carolina at Chapel Hill
E-mail: sujeong@cs.unc.edu, dm@cs.unc.edu

S. J. Guy
University of Minnesota
E-mail: sjguy@cs.umn.edu

Karl Hillesland
Advanced Micro Devices
E-mail: karl.hillesland@amd.com

B. Zafar
Hajj Research Institute, Umm Al-Qura University
E-mail: Bjzafar@uqu.edu.sa

A Gutub
Center of Research Excellence in Hajj and Omrah, Umm Al-Qura University
E-mail: aagutub@uqu.edu.sa

A preliminary version of this paper appeared in [16]

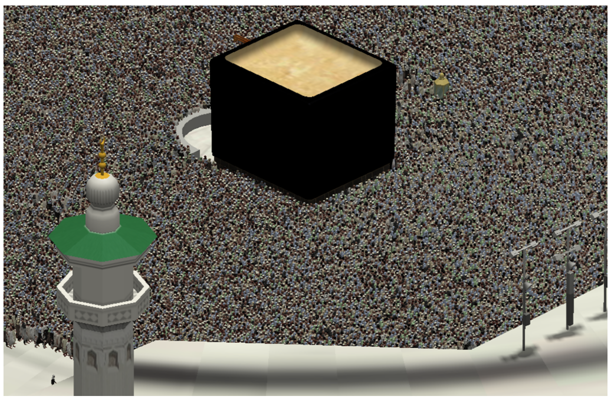


Fig. 1: **Simulation of Tawaf**: We simulate pilgrims performing the Tawaf ritual. In our scene, about 35,000 agents circle around the Kaaba, performing a short prayer at the starting line while some of the agents try to get towards the Black Stone at the eastern corner of Kaaba. We model the interactions between the agents in a dense crowd, such as when the agents are pushed by crowd forces (see video).

1 Introduction

Multi-agent simulations are frequently used to model a wide variety of natural and simulated behaviors, including human crowds, traffic, groups of birds, bees, fish, ants; 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. The challenge is to model these interactions in large multi-agent systems at interactive rates. Many multi-agent simulation techniques focus mainly on local navigation based on anticipatory collision avoidance and does not explicitly take into account physical interactions between agents or between agents and obstacles in the environment. Moreover, collision avoidance behavior towards the obstacles are limited to the static obstacles.

Methods which focus only on collision avoidance can work well in scenarios with low to medium density (e.g. less than 2 $agents/m^2$), where there is enough space for the agents to navigate freely without collisions. However, there are many situations such as political rallies, religious gatherings, or public subway stations, where agents can get very close to each other, and physical interactions between the agents frequently occurs. An agent may be pushed, or bumps into other agents in dense scenarios. For these kinds of dense crowds, it is important to model the direct physical interaction between the agents. Additionally, the indirect effect of the physical impact transferred to neighboring individuals in the crowd, such as the domino effect of people leaning against each other, may impact the trajectory of a high number of agents in a crowd. In extremely dense crowds, the forces from crowds sometimes become very large and can completely change the trajectory of an agent or make them fall. In these cases, crowd disasters can occur [38]. For all these reasons, understanding and simulating physical interaction between agents is necessary to simulate and analyze dense crowd scenarios. As the density of the crowd increases, it is more likely that even small motions can cause physical interactions with neighboring agents.

Similarly, the forces from many individual agents combine to produce a large effect on the environment. For example, crowds may push stacked boxes while moving through a narrow corridor and somebody may be hit by a falling boxes. 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 the obstacles in the 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 and avoid upcoming collisions, while also modeling physical responses to external forces in a single unified framework.

We compute the velocity of each agent as a linear programming problem in the velocity space. The resulting approach is efficient and can be used to simulate dense scenarios with thousands of agents at interactive rates. We further extend our method to model more complex behaviors involving social and cultural rules. We use finite state machines to define a series of behaviors as well as parameters for our physical interaction model. These parameters are used to distinguish between responsive collision avoidance behaviors and force-based physical interactions. For example, we use our approach to simulate various behaviors during the Tawaf ritual. We show that our velocity-based formulation can reliably simulate tens of thousands of agents in very dense scenarios (maximum density 8 $agents/m$), and model the physical interactions. Furthermore, we show that our approach is quite robust and we can use large time steps. We have also integrated our approach with the Bullet Physics Engine [1], and highlight the performance in many scenarios.

The rest of the paper is organized as follows. Section 2 gives a brief review of related work, Section 3 describe velocity-based physical interaction model combining anticipatory collision avoidance and physical forces, and Section 4 discuss high-level behavior modeling using a finite-state machine. We highlight the performance on different scenarios focusing on physical interactions in Section 6.

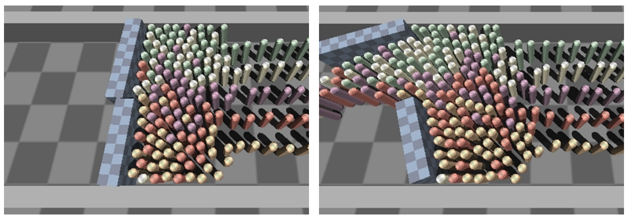


Fig. 2: **Wall Breaking**. We demonstrate the physical forces applied by cylindrical agents to breakable wall obstacles. Our algorithm can model such interactions between the agents and the obstacles in dense scenarios at interactive rates.

2 Related Work

2.1 Multi-Agent Motion 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 [30], which uses simple rules to model flocking behavior.

Force-based methods, such as the social force model [12], 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. Other approaches model collision-avoidance behavior with velocity-based techniques [3,29,14] or vision-based steering approaches [27].

Other techniques have been proposed to model complex social interaction. HiDAC [28] 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 [43] uses geometric proxies to model social priority, authority, guidance, and aggression. Many other multi-agent simulation algorithms exist, using techniques inspired by different fields such as sociology [25], biomechanics [9], and psychology [31,8,10,17] 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.

Other techniques use cognitive and decision-making models to generate human-like behaviors [33,44,41], or use data-driven approaches to the problem [20,22].

2.2 Dense Crowd Simulation

Density and crowd behaviors are closely related. The fundamental diagram is an empirically measured relationship between the pedestrian density and speed [32]. Some crowd simulation algorithms tend to adhere to the fundamental diagram. Curtis et al. [6] propose a method to simulate density-dependent behaviors for velocity-based collision avoidance technique. Lemercier et al. [21] focus on generating realistic following behaviors based on varying densities.

Other approaches for modeling crowds are based on continuum or macroscopic models [13,40,26]. In particular, Narain et al. [26] present a hybrid technique using continuum and discrete method for aggregate behaviors in large and dense crowds. These continuum methods are mainly used to simulate the macroscopic flow and may not model the detailed interactions between the individuals and the obstacles. In contrast, our approach simulates agent-agent and agent-obstacles physical interaction.

Some force-based techniques are used to simulate the interactions between agents in a dense crowd. Helbing et al. [11] model panic behavior with two additional physical forces (body force and sliding friction) in addition to the social forces. Yu and Johansson [45] propose a force-based technique to model the turbulence-like motion of a dense crowd by increasing the repulsive force.

2.3 Force-Based Techniques for Character Animation

There has been extensive work on using physics-based models to improve character animation. Sok et al. [37] use a force-based approach to ensure that the resulting motions are physically plausible. Other approaches consider geometric and kinematic constraints [36] or use interactive methods for character editing [15]. 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 [34,46]. Muico et al. [24] 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.

2.4 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. Other crowd simulation software such as Miarmy, Massive, and Golaem are integrated with modeling and rendering tools, and used to create character animation. 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 Velocity-based Modeling of Physical Interactions

Our approach extends the approach described in [16] to model the physical interactions between a large number of agents and obstacles. In this section, we give an

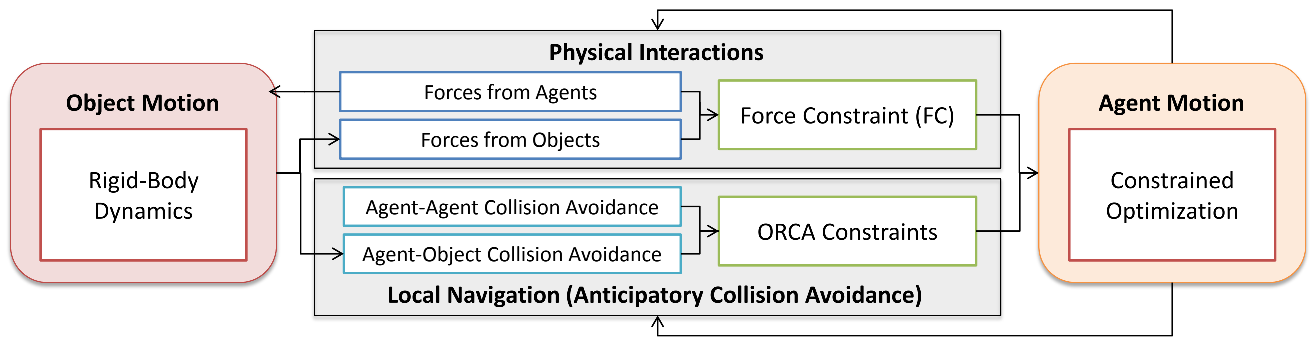


Fig. 3: **System Overview.** The motions for objects and agents are computed by a rigid-body dynamics solver and a constrained optimizer, respectively. Physical interactions between agents and obstacles determine forces. For obstacles, the forces serve as inputs to the rigid-body system; for agents, they become force constraints. These force constraints are combined with the original ORCA planning constraints and serve as inputs to optimization algorithm.

overview and a summary of techniques used to compute the forces.

3.1 Overview

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. Fig. 3 gives an overview of the full simulation system.

Agents are assumed to have a *preferred velocity*. This is the velocity at which the agent would travel if there were no collisions to avoid or physical forces acting on the agent. At each timestep, an agent computes a new velocity that satisfies the velocity constraints, then updates its position based on the preferred velocity. There are two types of constraints which we impose on an agent's velocity:

- **ORCA Constraints** guide the collision avoidance by specifying the space of velocities which are guaranteed to remain collision-free for a given period of time [3].
- **Force Constraints** account for forces which arise through physical interactions with other agents and objects.

Given an agent A with neighbors B, the *permitted velocities* for A, PV_A is the union of ORCA constraints and force constraints. We can state our agent update algorithm as an optimization problem. Formally:

$$PV_A = FC_A \cap \bigcap_{B \neq A} ORCA_{A|B}, \quad (1)$$

$$\mathbf{v}^{new} = \underset{\mathbf{v} \in PV_A}{\operatorname{argmin}} \|\mathbf{v} - \mathbf{v}^{pref}\|, \quad (2)$$

where $\mathbf{v}^{new}$, $\mathbf{v}^{pref}$ and FC_A are the new velocity, preferred velocity, and force constraints of A, respectively. $ORCA_{A|B}$ is ORCA constraints of A given its neighbors B.

3.2 Anticipatory Collision Avoidance

There are some significant differences between an agent's interaction with a neighboring agent and a dynamic obstacle, in terms of the motion computation. 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. We take account such difference into agent's collision avoidance behavior.

3.2.1 Agent-agent collision avoidance

ORCA constraints are defined by a set of velocities that are guaranteed to avoid upcoming collisions with other nearby agents. 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 **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:

$$ORCA_{A|B} = \{\mathbf{v}|(\mathbf{v} - (\mathbf{v}_A + \frac{1}{2}\mathbf{u})) \cdot \hat{\mathbf{u}} \geq 0\}, \tag{3}$$

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.

3.2.2 Agent-dynamic obstacle collision avoidance

The dynamic object O is represented, like the agents, as an open disc that is a 2D projection of the bounding sphere of the object. We use this bounding shape for collision avoidance since the agent's navigation is performed in 2D space, but the underlying rigid body simulation uses an 3D object shape for handling collisions with other rigid bodies in the scene.

Agents try to avoid collisions with dynamic obstacles 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 3.2.1) is required to avoid an anticipated collision with an obstacle, the collision avoidance constraint for agent A induced by object O is:

$$ORCA^{\tau}_{A|O} = \{\mathbf{v}|(\mathbf{v} - (\mathbf{v}_A + \mathbf{u})) \cdot \hat{\mathbf{u}} \geq 0\}. \tag{4}$$

3.3 Constraints from Physical Forces

We give brief description of the forces in this section. *Contact forces* include pushing forces and collision forces, and model collision response force or an attempted pushing. *inferred forces* include deceleration force and resistive force, and model the impact of forces on agent's motion. For more detail, please refer to [16].

3.3.1 Force computation

Pushing Forces: Pushing is one of the ways for agents to *physically* interact with each other [28]. In our formulation, the pushing force $\mathbf{f}^p_{i|k}$ exerted by an agent i pushing another agent k can be given as:

$$\mathbf{f}^p_{i|k} = \rho_k f_p \frac{\mathbf{p}_k - \mathbf{p}^+_i}{\|\mathbf{p}_k - \mathbf{p}^+_i\|}, \tag{5}$$

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 a magnitude of total pushing force of agent i towards all interacting agents. It can be defined by the designer, but in our examples, we compute this value to be proportional to agent i's current speed. ρ_k is used to define the weight of pushing force towards each interacting agent k. For our examples, 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. This force is computed based on the physically-based simulation approach proposed by [2]. We consider only linear momentum and simulate agents as radially symmetric disks. For an agent i colliding with agent k, the collision force $\mathbf{f}^c$ is computed as follows:

$$\mathbf{f}^c = (\frac{-(1+\epsilon)\mathbf{v}^{rel}}{1/m_i + 1/m_k} \cdot \mathbf{n})/\Delta t, \tag{6}$$

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. In our examples, we assign a uniform mass to each agent, but any reasonable mass value can be used for the simulation.

In case of a collision between an agent and a dynamic object, the impulse force is computed in the same way. 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:

$$\tau^c = \mathbf{f}^c \times \mathbf{r}_o, \tag{7}$$

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

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:

$$\mathbf{f}^d_i = \begin{cases} k_{thresh} m_i \Delta\mathbf{v}_i/\Delta t & \text{if } (\Delta\hat{\mathbf{v}}_i \cdot \hat{\mathbf{v}}_i) < -cos(\theta_d), \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

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. Agents can absorb or transform forces, which are approximated by 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 is no interacting agent, we assume it is the former case, and the deceleration force is applied back to the agent itself. In the latter case, we distribute the deceleration force among the neighbors. 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.

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 (2) 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:

$$\mathbf{f}_i^r = \begin{cases} k_{thresh} m_i(\mathbf{v}_i - \mathbf{v}_i^f)/\Delta t & \text{if } \mathbf{v}_i^f \neq \mathbf{0} \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

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.

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.

3.3.2 Force constraints

The net force $\mathbf{f}$ is the sum of all the forces applied to the agent:

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

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 $\hat{\mathbf{f}}$. It contains a set of velocities that is equal to or greater than the minimum velocity change required by the force $\mathbf{f}$.

3.4 Benefits of Force Constraints

By introducing inferred forces, our method can model balance recovery motions that cannot be captured by physics-based rigid body dynamics.

Balance recovery motion: When forces are applied, rigid body motion changes accordingly to the Newtonian dynamics. However, humans have the ability to absorb and resist the external forces even from unexpected events such as sudden pushes or an impact from an obstacle. In these situations, humans take effort to keep their balance creating a behavior known as *balance recovery* in Biomechanics [23]. Balance recovery is important to model human locomotion, and has been studied in other fields like robotics for humanoid robots [42] and in computer graphics for animated characters [35].

Typical balance recovery motions include taking additional steps or reaching and grasping an object for support. When humans fails to recover the balance, they take further adjustments to refine their initial responses. In other words, the balance recovery can affect the motion, including the trajectory, for a period of time. The balance recovery motion is a result of both physical and cognitive activity, which also depends on the environmental constraints and affordances (e.g., space to step, objects to grasp for support) [23].

We define two forces, deceleration force and resistive force that are used to simulate the behavior corresponding to balance recovery. We infer these forces from the agent's motion at a given time, based on our assumption that the motion of an agent can be decomposed into two components: collision avoidance and Newtonian dynamics. Loosely speaking, we treat the different between the velocity implied by the physics forces and the resultant velocity produced by the simulation as a recovery force which is applied on the nearby environment and agents.

Force propagation: Forces applied to an agent can propagate through a dense crowd, since one agent is likely to exert forces on others for support in order to recover from the external pushing force. The propagation forces can be inferred when the motion computed using constrained optimization does not match the motion expected from external physical forces. In this case, we assume that the agent's action of balance recovery took place to resist the 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, while resist the effect of the forces. These inferred propagation forces are applied to the appropriate neighboring agents during the subsequent timestep.

4 Higher-Level Behavior Modeling

In many cases, the crowd or individual behaviors change over a period of time. Cultural, social norms, as well as personal goals and intentions can change a person's behaviors over time. Likewise, individuals can exhibit role-specific behaviors, in a variety of situations which can fundamentally change how they interact with each other. A clear example of this can be seen in sporting events, where the behavior towards the players of their own team members are cooperative (and avoid collisions), whereas the behavior towards members of the opposite team often includes blocking, tackling and other forms of physical collisions. Ideally, we would like to model a full variety of such behaviors using our physically-based interactions, while accounting for changes in behavior.

Incorporating such a variety of behaviors changing over time, and depending on the situation or social, cultural norms requires a way to model higher-level decision making process and behavior rules. A common approach to achieve state-dependent behaviors in general is to use Finite State Machines (FSM). In this section, we show how our simulation approach can be combined with FSMs to model such complex physical behaviors in multi-agent simulations. The resulting framework provides a natural way to define different behavior patterns and changes of those behaviors over time and in different situations.

4.1 The Behavior Finite State Machine

An FSM is a machine or a model which has finite number of state and transitions between the states. FSMs have been widely used as a way to model intention and decision making process for agents in Crowd Simulation and Games [39,4]. For example, FSMs can describe a set of behaviors for an agent with certain social status (e.g., a leader) along with the transition of these behaviors in certain situations (e.g., safe state or dangerous state) based on the leader agent's perceived information. Recent work has integrated such an FSM-based behavior specification with velocity-based local collision avoidance schemes to simulate crowds displaying various behaviors [5]. We present an improved algorithm that extends this framework to produce complex simulations with physically-based agent interactions.

Fig. 4 shows the overall architecture of the FSM based behavior modeling for our physical interaction model. We specify a set of actions (behaviors) for each state, along with interaction parameters that change the behavior of the individual agents and the crowd. Transitions between the FSM states are made based on the result of our physical interaction model combined with local collision avoidance method. This corresponds to the decision making process of an agent, which is based on the perceived information about the agent itself and its neighbors. The perceived information and the decision of current action altogether are used to compute agents' local navigation planning. In other words, we model agents behavior using FSM by specifying the interaction parameters that define perception and local planning.

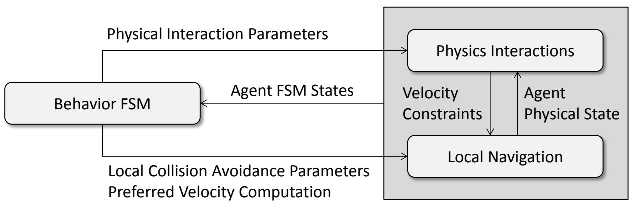


Fig. 4: **Overview of FSM-based behavior modeling** FSM states are used to specify a set of available actions, along with the parameters used for physics-based interaction and local collision avoidance. Transitions between the FSM states are made based on the result of our physical interaction model combined with local collision avoidance method.

Importantly, it's the combined interplay between an agent's perceived information and its behavioral simulation parameters that determine its actions in a simulation. For example, if we reduce an agent's perception by only allowing it to sense very close neighbors, the agent will be less able to plan ahead to avoid collisions and more likely to run into neighbors. However, when the two agents collide, the magnitude of the interaction will be controlled by the physical interaction parameters. Therefore, a rude or hurried agent state can be created by reducing the perception range, and increasing the pushing forces, and a more polite or relaxed agent state can by made by increasing the perception range and decreasing pushing. In this way, we use the same motion model (Eq. 1 and Eq. 2) across all the states of the FSM, but leverage the agent parameters to diversify the agent interactions.

Because of this relationship between collision avoidance parameters and physical interaction parameters, both should be considered together when designing the FSM. Table 1 shows what both of these components should be like in terms of intentional behaviors and responsive behaviors. For intentional interactions, the agent's primary goal is to approach the target and apply intended forces. In this case, higher anticipatory collision avoidance behavior prevents the agent from getting closer to the target. Rather, the agent should be able to approach the target even when they perceive impending collision with the target. For example, a soccer player would even run towards the ball even when the ball is approaching the player at a high speed. To model such behavior, local collision avoidance behavior should be minimized to allow the agent to physically interact with the target, but still be able to prevent overlap with the target object.

	Intentional physical interaction	Responsive behaviors
Preferred velocity	towards the target	(local, global) destination
Collision avoidance	no overlap	anticipated collision avoidance
Physical interaction	applying forces (varying magnitude and direction)	collision response or balance recovery

Table 1: Different interaction parameters for intentional behaviors and responsive behaviors

5 Results

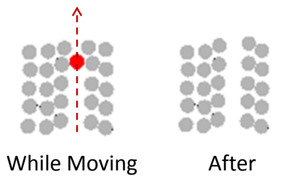
In this section, we highlight the performance of our algorithm in different scenarios. We first show some results of our physical interaction model, and then present FSM-based behavior modeling with dodge-ball game scenario and Tawaf ritual scenario. We also analyze the approach and compare it with other techniques. We direct the readers to the video or the preliminary version paper [16] for more results on physical interaction models.

5.1 Agent-Agent Interaction

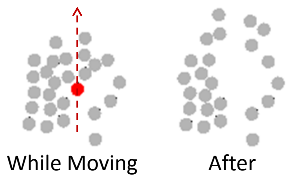
We demonstrate a few scenarios which highlight the effect of physical interactions between agents and how those effects propagate through crowds.

Running Through Scenario: We demonstrate a scenario where an agent runs at a high speed and push through a dense crowd of 25 agents that are standing still. Fig. 5 compares the result of our method to those achieved using multi-agent simulation without any physical interactions.

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. 5 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.



(a) without physical interactions



(b) with physical interactions

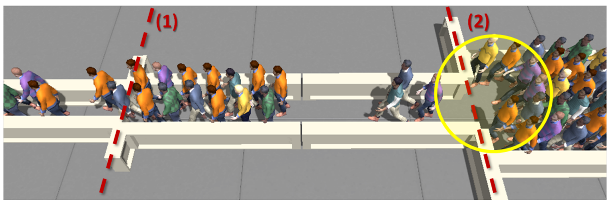
Fig. 5: **Rushing through still agents**: The red agent tries to rush through a group of standing agents, simulated (a) with only anticipatory collision avoidance and (b) with physical interactions. Using our method, the forces are propagated among the agents, resulting in a new distribution pattern (b).

Two Bottlenecks Scenario: In this scenario, long lines of closely spaced walking agents attempt to pass through two narrow bottlenecks, as illustrated in Fig. 6. The first bottleneck (denoted in the figure as (2)) is about the width of two agents; the second is narrower, about wide enough for one agent (denoted 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. 6 (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.

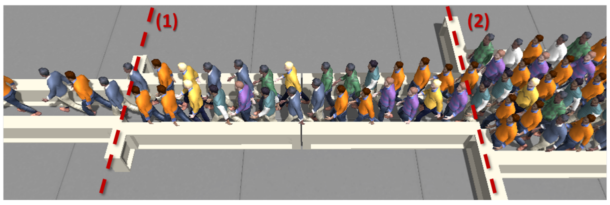
5.2 Agent-object Interaction

We also demonstrate the effect of forces between dynamic objects and agents. We used the Bullet Physics engine [1] to compute the motion of dynamic obstacles (3D rigid body dynamics). The results 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.



(a) Multi-agent simulation with no physical interaction

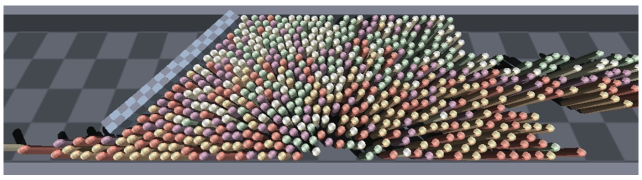


(b) Physical interaction amongst agents and with the walls

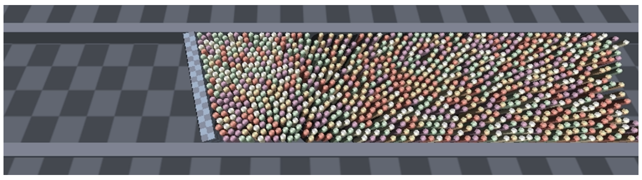
Fig. 6: **Two bottlenecks scenario** We simulate and compare crowd behavior at two narrow bottlenecks, which are marked with red dotted lines. Bottleneck (1) is barely wide enough for one person to pass through; bottleneck (2) is about twice that width and allows two agents to pass through it at a time. 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.

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. 2 shows stills from the simulation.

Changing the various properties of the wall changes how the crowd interacts. Fig. 7 shows the result of simulation with two different configurations of the wall. In the first configuration, when the blocks are tightly attached, the wall is not broken. Instead, it is moved and rotated by crowd forces, and makes a gap for the crowd to escape through. In the second configuration, when the wall consists of much heavier blocks that are glued together tightly, it does not break or move easily even after the crowd (1200 agents) has entered the isle. In this configuration, the crowd sometimes makes a wave-like movement where the sparse density crowd movement is propagated from front to back and vice versa.



(a) Tightly attached wall blocks



(b) Tightly attached heavier blocks (zoom out view)

Fig. 7: **Wall Breaking Simulations with Different Wall Properties**. (a) When the blocks are tightly attached, the wall is not broken. Instead, it is moved and rotated by crowd forces, and made a gap for the crowd to escape through. (b) When the wall consists of much heavier blocks, it does not break or move easily even after all the crowds (1200 agents) entered the isle. In this example, sometimes crowd makes a wave-like movement where sparse density crowd movement is propagated front to back and vice versa.

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. 8 shows a still from the simulation.

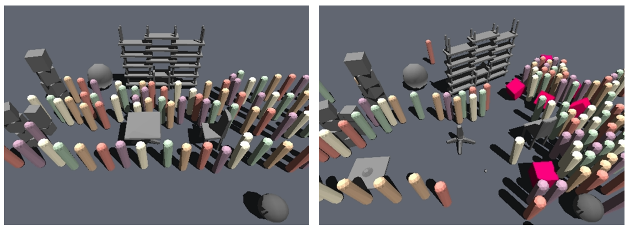


Fig. 8: **Office Scenario**. Agents navigate to avoid office furniture. As users insert flying pink boxes into the scene, the agents get pushed, collide into each other, and avoid falling objects (see video).

5.3 Dodge-ball scenario

As an example of state-based, physical interaction, we show how our FSM based algorithm can be used to

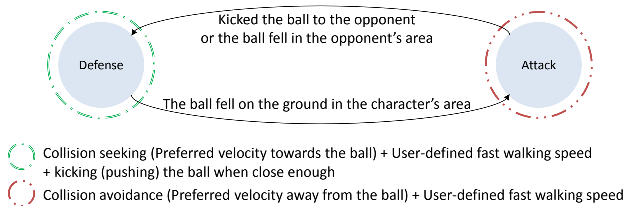
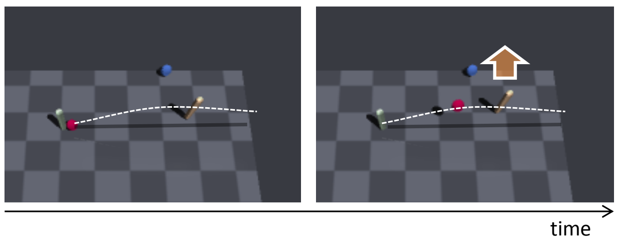
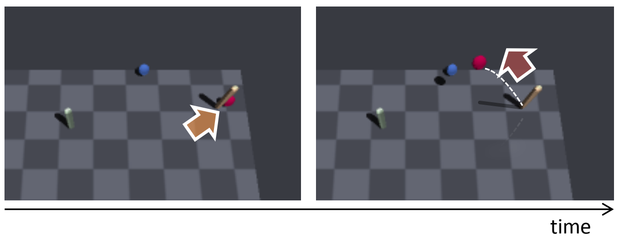

Kicked the ball to the opponent
or the ball fell in the opponent's area
Defense
Attack
The ball fell on the ground in the character's area
Collision seeking (Preferred velocity towards the ball) + User-defined fast walking speed + kicking (pushing) the ball when close enough
Collision avoidance (Preferred velocity away from the ball) + User-defined fast walking speed

Fig. 9: **FSM for Dodge-ball Scenario** We use a simple two-state FSM to specify the game rule. The states consists of defense state and an attack state. During the attack-state, the character chases a ball and kicks it to its opponent. A character in defense-state tries to avoid the ball until the ball fell down on the ground. State transitions occur based on the location of the ball and the kicking (applying force to the ball) action performed by the character.



(a) Orange character avoids a ball



(b) Orange character approaches and kicks a ball

Fig. 10: **Behavior examples modeled by our method** (a) Green character (user-control, left) kicks the red ball to the orange character (computer control, right). Orange character is in the defense-state at that moment, and tries to avoid the ball. (b) When the ball falls down, the orange character's state is changed to the attack-state. Collision avoidance behavior is changed just to meet non-overlapping condition with the ball and the character's preferred velocity is updated towards the ball. The orange character approaches the ball and kicks the ball to the green character.

control behavior changes in simulated game of dodge ball. Here, we created an interactive dodge-ball game, where a user can control one of the game characters, and the computer program controls the other character(s). A two-state FSM is used to specify the behaviors (See Fig. 9), with the states consisting of a defense state and an attack state. During the attack state, a character chases a ball and kicks it to its opponent. A character in defense-state tries to avoid the ball until the ball rolls on the ground. State transitions occur based on the location of the ball and the kicking action performed by the characters.

Figure 10 shows part of the scenario highlighting the change in interaction between the agent and the ball. The first two images, Fig. 10(a), show the agent behavior in the defense-state. The user-controlled green character on the left kicks the red ball at the FSM-controlled orange character on the right. Initially the red character tries hard to avoid the ball, with the local collision avoidance algorithm for this character considering a large perception radius, with a long time duration, when computing its motion. In cases where the agent is not able to avoid the collision (e.g. the ball is moving too fast), there is a physically simulated collision response between the ball and the agent. Due to relatively smaller mass (0.4kg) of the ball compared to the characters (70kg), the effect of collision and resulting forces is much larger on the ball. If the FSM-agent is successful in avoiding the collision, its state will change to the attack state (Fig. 10). Here, the agent's collision avoidance behavior is changed allowing it to approach the ball as fast as possible (e.g., small sight radius and large preferred velocity), then to kick the ball towards the green character. The force applied to the ball is computed based on the speed of the character, and its direction is towards the user-controlled agent.

As can be seen in this example, our overall approach can model collision avoidance, collision response, applying intentional forces, and decision making for the character (e.g., goal position and transition of the state). In the supplementary video, we show an expanded version of this scenario with increased number of balls. We can observe a character chasing a ball while avoiding other dynamic obstacles, and, at times, pushing through them to attack the other character. Properly simulating these interacting behaviors requires physical interaction, local collision avoidance, and behavioral states to be combined together in the same framework as we have presented here.

5.4 Large-Scale Simulation: Tawaf Scenario

Because our method has only a small computational cost per-agent, and is stable even in dense scenarios, it can be used to produce complex, large-scale simulations with agents physically interacting across a variety of different behaviors. To illustrate this, we performed a case-study in simulating the large, dense crowd performing the Islamic ritual of pilgrimage called the Tawaf.

During the Tawaf, pilgrims walk in a circle around the Kaaba, the large central structure, seven times counter-clockwise for prayers. While circulating, many pilgrims try as part of the ritual to reach the central black stone located at the eastern corner of the Kaaba; alternatively, they perform a short prayer while facing the Kaaba at the beginning of each circuit. The walkable area surrounding the Kaaba, is known as the Mataf and can support upwards of 35,000 pilgrims gather to perform the Tawaf [5].

We have extended the simulation of Curtis et al. [5], to produce a simulation of the Tawaf ritual with about 35,000 FSM-driven, physically-interacting agents as described in Section 4.

The FSM we use, both sets the goals of the agents in order to follow the steps of the ritual and modifies the agent's behavioral parameters to help achieve these goals. For example, some of the agents will probabilistically choose to move closer to the central Kaaba structure in order to approach a religiously significant black stone. Agent in the "Move to Black Stone" state are allowed to exert physical pushing forces on their nearby neighbors in order to successfully move through the dense crowd to reach the stone. Figure 11 shows all the FSM states and transition conditions, along with the descriptions about a few important variables conditions for the local navigation and physical interactions for the corresponding state. Most notably, agents exert pushing forces on the crowd if they are trying to touch to the black stone on the Kaaba or are trying to exit the Mataf after completing the ritual.

Flow Analysis: We measured the average speed and density of the agents from our simulation. First, we computed the average speed in different regions of Mataf shown in [5]; there is an overall trend towards higher speed in region 6 and towards a lower speed in regions 1 and 7 when compared to the speeds of their neighboring regions (see Fig. 14). These highest speed and lowest speed regions also match with the real world data provided by [18]. Second, we computed the density in different regions of Mataf based on our Tawaf simulator. Empirically, the density on the Mataf floor can be as high as 8 $people/m^2$ [5]; our method gives a maximum density around 7.4 $agents/m^2$ (see Fig.12).

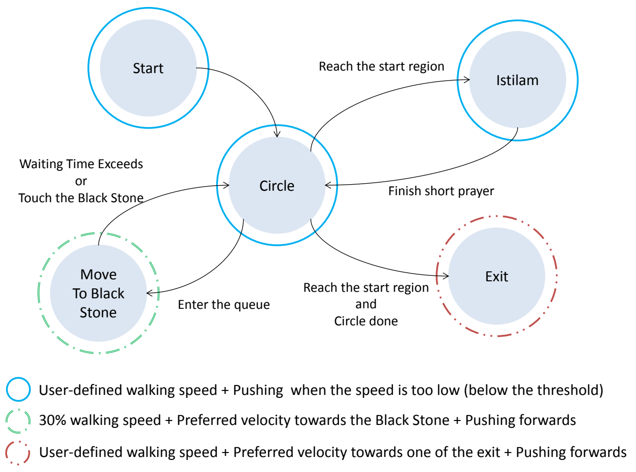

Start
Reach the start region
Istilam
Waiting Time Exceeds
or
Touch the Black Stone
Circle
Finish short prayer
Move
To Black
Stone
Enter the queue
Reach the start region
and
Circle done
Exit
User-defined walking speed + Pushing when the speed is too low (below the threshold)
30% walking speed + Preferred velocity towards the Black Stone + Pushing forwards
User-defined walking speed + Preferred velocity towards one of the exit + Pushing forwards

Fig. 11: **Agent States and Transitions**. The Tawaf states are represented as blue circles and transition condition between these states are marked with arrows. We associate different properties like walking speed, pushing condition, etc., with the agent behavior.

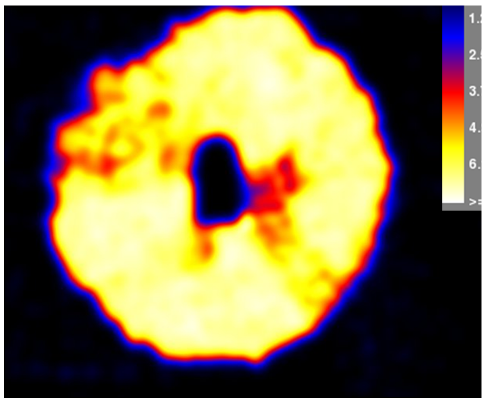
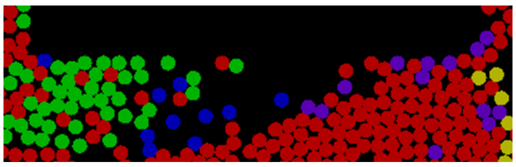


Fig. 12: **Density from the Simulated Result**. Reported densities on the Mataf floor can be as high as 8 $people/m^2$ [5]; our method gives a maximum density around 7.4 $agents/m^2$

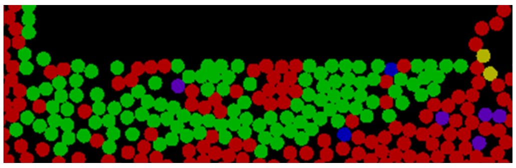
5.4.1 Effect of physical interactions

We perform two experiments to show the benefit of physical interactions in large, dense crowd settings. The first experiment shows crowd forces acting on agents. The second experiment compares the overall crowd flow simulated during the Tawaf ritual under increased pushing behaviors between agents.

Pushing in the Queue to the Black Stone : As part of the Tawaf, we simulate the movement of pilgrims waiting in lines to touch or kiss the Black Stone (the eastern cornerstone of Kaaba). Pilgrims in this region makes distinctively slow motion patterns compared to the other pilgrims circling around the Kaaba. After they



(a) Without Physical Interaction



(b) With Physical Interaction

Fig. 13: **Pushed by crowd**. Green circles represent the agents in the queue waiting to touch the black stone. The green agents slow down and the result in heavy congestion at the beginning of the queue. Without physical interactions, agents are stuck in the beginning of the queue although there is a space in front of the queue to proceed. By adding physical interactions, the agents in the queue are pushed by the crowds, and move towards the black stone without breaking the queue.

touch the Black Stone, these pilgrims join the rest of circling flow and adjust to the speed of other neighbors.

In the real-world video (see the supplementary video), we can observe that some of the pilgrims in the queue are often pushed by neighboring pilgrims. We attempted to simulate such crowd force that are applied to the agents in a dense crowd. To see the effect of crowd force, we assign lower preferred speeds to the agents who have entered the beginning of the queue. Figure 13 shows the 2D comparison between the simulated queuing behavior for the Black Stone, (a) when no physical interactions added and (b) with physical interactions. Green circles represent the agents in the queue, red circles represent the agents circling around the Kaaba, blue circles represent the agents leaving the queue and start circling after given waiting time.

Due to the sudden slowdown caused by the green agents, heavy congestion is made at the beginning of the queue. Without adding physical interactions, we cannot capture the effect of crowd force applied to these agents even in such high density. Agents are stuck in the beginning of the queue although there is a space in front of the queue to proceed. By adding physical interactions, the agents in the queue are pushed by the crowds, and move towards the black stone without breaking the queue.

Pushing Towards Exits : We also evaluate the effect of physical interactions in large crowds. First, we compare the average speed of the agents with and

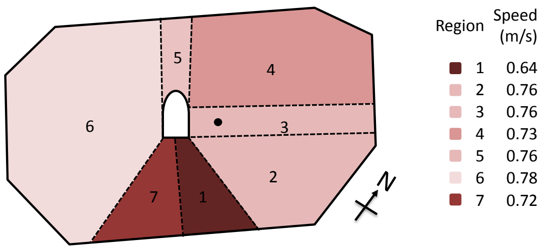


Fig. 14: **Region Speed from the Simulation Result** Average speed of each region of the Mataf area. It matches the overall trend corresponding to higher average speed (region 6) and lower speeds (regions 1 and 6) observed by [18].

without physical interactions. Fig. 14 shows the average speed of the agents measured in the several different regions as proposed in [18]. The overall trend of relative speeds between regions is the same with and without pushing, but the average speed increase as agents push more. In both cases, the trends match well with those reported in the Tawaf literature [18].

Additionally, we run the same scenario with increased number of aggressive pushing agents. When an agent finishes the ritual, the agent is assigned a randomly selected exit (from the five exits in the Mataf area) as their goal position, and tries to push through the crowd. Since the exiting agents have to escape through a very dense crowd while also moving in the circular flow, their pushing forces affects the average speed of entire region. At any given point in the simulation, about 2% of the total number of agents are trying to exit. By adding more pushing agents, the average speed increased about $0.2m/s$. Figure 15 summarizes the speed of the agents in each of the Tawaf regions.

6 Analysis

Our approach is mainly designed for interactive applications that require plausible physical behavior (e.g. games or virtual worlds) as well as real-world scenarios with high crowd densities. By using a combination of force and navigation constraints that affect agents' behavior, our approach can simulate many useful 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

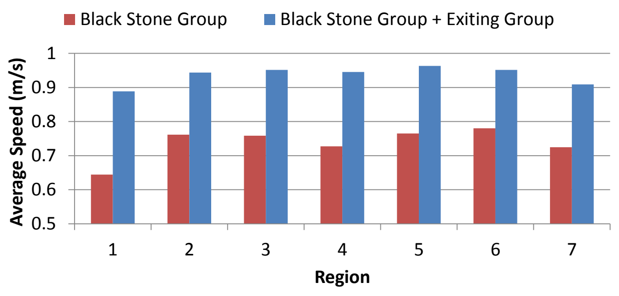


Fig. 15: **Comparison of average region speeds**. Blue bars correspond the average speeds of the agents in each region when we introduce excessive pushing behavior to the exiting agent and queuing agents. Red bars correspond the average speeds when only the queuing agents pushes forwards while moving towards the Black Stone. Increasing number of pushing behaviors brought a 20% to 40% increase in average speed.

paradigms of forces and navigation into a unified framework and compute the new velocity for each agent using linear programming.

Performance: We measured the simulation timings for the demos we presented in earlier sections (see Table 2). 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	Num. Agents	Dynam. Obsts.	Static Obsts.	fps
Two Bottlenecks	1000	0	20	829.7
Wall Breaking	1200	200	2	50.1
Office	1200	65	0	69.0
Dodge Ball	2	500	4	90.9
Tawaf Sim.	35000	0	23	5.7

Table 2: Performance on a single core for different scenarios. Our algorithm can handle all of them at interactive rates.

Stability Analysis: It is well known that many forced based simulation models, such as social force models commonly used to simulate crowd (e.g. [12]) are prone to stability problems that can even occur at small step sizes [19]. These problems include oscillation and loss of accuracy in terms of trajectory computation. Using bigger time steps can make the problem worse. In contrast, velocity based collision avoidance techniques have been shown to produce stable simulations in large, dense crowds [7]. Our approach preserves this stability across timesteps while still accounting for physical forces.

One way to analyze the stability of our approach is by analyzing the number of times we are unable to find a feasible velocity that satisfies all the constraints, e.g. both anticipated collision avoidance constraints and force constraints. When this occurs, it means an agent can choose a potentially colliding velocity or is not respecting the physical constraints.

We perform a test using a scenario similar to Fig. 5. An aggressive agent pushes through 50 standing agents, and the pushed agents sequentially exert forces on adjacent neighbors by generating physical interactions. During the simulation, we measure the number of times when linear optimization fails to find the feasible velocity satisfying all the constraints. In this case, we perform higher-order optimization to find a solution.

As can be seen in Fig. 16, at the peak of congestion in the simulation 95% of the agents are able to find velocities that satisfy all their constraints. Importantly, this stability holds across a variety of timesteps. As the timestep size varies from 0.01 up to 0.2, most of the agents are still able to find constraint-satisfying velocities. In general, the behavior does not change significantly across this wide range of time steps.

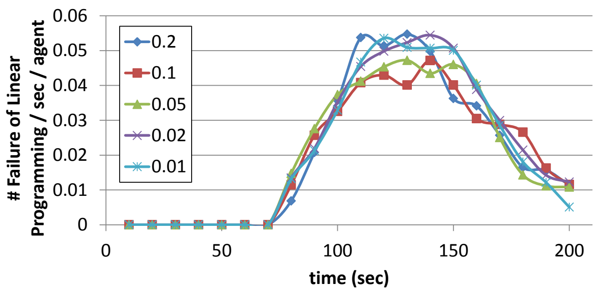


Fig. 16: **Number of constraint optimization failures**. We analyze the stability of our method by measuring the number of constraint optimization failures. At the peak of congestion in the simulation, 95% of the agents are able to find velocities that satisfy all their constraints. Importantly, this stability holds across a variety of timesteps. As the timestep size varies from 0.01 up to 0.2, most of the agents are still able to find constraint-satisfying velocities.

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 [12] use forces to model social factors (e.g. attraction and repulsion) between the agents, not physical interactions. Most closely related to our work are methods such as [11,45,28]. These methods 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.

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 [5,19]. Our method provides stable anticipatory motion for agents while incorporating agent responses to forces. It can be easily combined with other velocity-based approaches.

In terms of large, dense crowd simulation, continuum-based methods such as [13,40] or hybrid method coupling continuum-based method and velocity-based method [26] can be effective solutions. However, these methods do not model physical interactions between the agents or obstacles. Moreover, it is hard to extend these methods to model individually varying behaviors or high-level social behaviors.

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 disc). This means that we cannot accurately simulate physical interactions with human-like articulated models and 3D objects.

7 Conclusions 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. We also showed that our approach can be extended to model more complex behaviors involving a decision making process. In addition, we simulated real world examples of massive crowds such as in the Tawaf ritual. Our method was able to generate many emergent behaviors compared with real-world behaviors.

As future work, we would like to further explore our method by comparing the results with real-world crowd behaviors and performing more validation. Moreover, in many scenarios, the external forces could change an agents behavior. For example, applying intentional forces such as pushing, kicking can slow down an agent. Such a phenomena could be well incorporated in our approach given studies about how such forces can limit human behavior (e.g., from biomechanics). Furthermore, we need better techniques to collect data about real-world crowds in dense settings and use them to validate the simulation algorithms. We would also like 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.

References

1. AMD: Bullet Physics 2.80. http://bulletphysics.org (2012)
2. Baraff, D.: 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, p. 97 (1997)
3. van den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Robotics Research: 14th ISRR (STAR), vol. 70, pp. 3–19 (2011)
4. Buckland, M.: Programming game AI by example. Jones & Bartlett Learning (2005)
5. Curtis, S., Guy, S.J., Zafar, B., Manocha, D.: 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, pp. 128–135 (2011)
6. Curtis, S., Manocha, D.: Pedestrian simulation using geometric reasoning in velocity space. In: Proceedings of Pedestrian and Evacuation Dynamics (2012)
7. Curtis, S., Snape, J., Manocha, D.: Way portals: efficient multi-agent navigation with line-segment goals. In: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, I3D '12, pp. 15–22. ACM, New York, NY, USA (2012)
8. Durupinar, F., Pelechano, N., Allbeck, J., Gü anddü and-kbay, U., Badler, N.: How the ocean personality model affects the perception of crowds. Computer Graphics and Applications, IEEE **31**(3), 22 –31 (2011)
9. Guy, S.J., Curtis, S., Lin, M.C., Manocha, D.: Least-effort trajectories lead to emergent crowd behaviors. Phys. Rev. E **85**, 016,110 (2012)

10. Guy, S.J., Kim, S., Lin, M.C., Manocha, D.: Simulating heterogeneous crowd behaviors using personality trait theory. In: Symposium on Computer Animation, pp. 43–52. ACM (2011)
11. Helbing, D., Farkas, I., Vicsek, T.: Simulating dynamical features of escape panic. Nature **407**(6803), 487–490 (2000)
12. Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Phys. Rev. E **51**, 4282–4286 (1995)
13. Hughes, R.L.: The flow of human crowds. Annual Review of Fluid Mechanics **35**(1), 169–182 (2003)
14. Karamouzas, I., Overmars, M.: Simulating and evaluating the local behavior of small pedestrian groups. IEEE Trans. on Visualization and Computer Graphics **18**(3), 394–406 (2012)
15. Kim, M., Hyun, K., Kim, J., Lee, J.: Synchronized multi-character motion editing. ACM Trans. Graph. **28**(3), 79:1–79:9 (2009)
16. Kim, S., Guy, S.J., Manocha, D.: Velocity-based modeling of physical interactions in multi-agent simulations. In: Eurographics/ ACM SIGGRAPH Symposium on Computer Animation. Eurographics Association (2013)
17. Kim, S., Guy, S.J., Manocha, D., Lin, M.C.: Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory. In: Symposium on Interactive 3D Graphics, I3D '12, pp. 55–62. ACM, New York, NY, USA (2012)
18. Koshak, N., Fouda, A.: Analyzing pedestrian movement in mataf using gps and gis to support space redesign. In: The 9th International Conference on Design and Decision Support Systems in Architecture and Urban Planning (2008)
19. Köster, G., Treml, F., Gödel, M.: Avoiding numerical pitfalls in social force models. Phys. Rev. E **87**, 063,305 (2013). DOI 10.1103/PhysRevE.87.063305. URL `http://link.aps.org/doi/10.1103/PhysRevE.87.063305`
20. Lee, K.H., Choi, M.G., Hong, Q., Lee, J.: Group behavior from video: a data-driven approach to crowd simulation. In: Symposium on Computer Animation, pp. 109–118 (2007)
21. Lemercier, S., Jelic, A., Kulpa, R., Hua, J., Fehrenbach, J., Degond, P., Appert-Rolland, C., Donikian, S., Pettré, J.: Realistic following behaviors for crowd simulation. Computer Graphics Forum **31**(2pt2), 489–498 (2012)
22. Lerner, A., Chrysanthou, Y., Shamir, A., Cohen-Or, D.: Data driven evaluation of crowds. In: MIG, pp. 75–83 (2009)
23. Maki, B., McIlroy, W., Fernie, G.: Change-in-support reactions for balance recovery. Engineering in Medicine and Biology Magazine, IEEE **22**(2), 20–26 (2003)
24. Muico, U., Popović, J., Popović, Z.: Composite control of physically simulated characters. ACM Transactions on Graphics **30**(3) (2011)
25. Musse, S.R., Thalmann, D.: A model of human crowd behavior: Group inter-relationship and collision detection analysis. In: Proc. Workshop of Computer Animation and Simulation of Eurographics'97, pp. 39–51 (1997)
26. Narain, R., Golas, A., Curtis, S., Lin, M.C.: Aggregate dynamics for dense crowd simulation. ACM Trans. Graph. **28**(5), 122:1–122:8 (2009)
27. Ondřej, J., Pettré, J., Olivier, A.H., Donikian, S.: A synthetic-vision based steering approach for crowd simulation. ACM Trans. Graph. **29**(4), 123:1–123:9 (2010)
28. Pelechano, N., Allbeck, J.M., Badler, N.I.: Controlling individual agents in high-density crowd simulation. In: Symposium on Computer animation, pp. 99–108 (2007)
29. Pettré, J., Ondřej, J., Olivier, A.H., Cretual, A., Donikian, S.: Experiment-based modeling, simulation and validation of interactions between virtual walkers. In: Symposium on Computer Animation, SCA '09, pp. 189–198. ACM (2009)
30. Reynolds, C.: Steering Behaviors for Autonomous Characters. In: Game Developers Conference 1999 (1999)
31. Sakuma, T., Mukai, T., Kuriyama, S.: Psychological model for animating crowded pedestrians: Virtual humans and social agents. Comput. Animat. Virtual Worlds **16**, 343–351 (2005)
32. Seyfried, A., Steffen, B., Klingsch, W., Boltes, M.: The fundamental diagram of pedestrian movement revisited. J. Stat. Mech. **2005**, P10,002 (2005)
33. Shao, W., Terzopoulos, D.: Autonomous pedestrians. In: Symposium on Computer animation, pp. 19–28 (2005)
34. Shapiro, A., Pighin, F., Faloutsos, P.: Hybrid control for interactive character animation. In: Pacific Conference on Computer Graphics and Applications, PG '03, pp. 455–. IEEE Computer Society, Washington, DC, USA (2003)
35. Shiratori, T., Coley, B., Cham, R., Hodgins, J.K.: Simulating balance recovery responses to trips based on biomechanical principles. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2009)
36. Shum, H.P.H., Komura, T., Yamazaki, S.: Simulating multiple character interactions with collaborative and adversarial goals. IEEE Transactions on Visualization and Computer Graphics **18**(5), 741–752 (2012)
37. Sok, K.W., Yamane, K., Lee, J., Hodgins, J.: Editing dynamic human motions via momentum and force. In: Symposium on Computer Animation, SCA '10, pp. 11–20. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2010)
38. Still, G.K.: Proximate and distal causality `http://www.gkstill.com/ExpertWitness/CrowdDisasters.html` (2013)
39. Thalmann, D., Musse, S.: Behavioral animation of crowds. In: Crowd Simulation, pp. 111–168. Springer London (2013). DOI 10.1007/978-1-4471-4450-2_5. URL `http://dx.doi.org/10.1007/978-1-4471-4450-2_5`
40. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. In: ACM SIGGRAPH 2006, pp. 1160–1168. ACM (2006)
41. Ulicny, B., Thalmann, D.: Towards interactive real-time crowd behavior simulation. In: Computer Graphics Forum, vol. 21, pp. 767–775. Wiley Online Library (2002)
42. Wei, Y., Gang, B., Zuwen, W.: Balance recovery for humanoid robot in the presence of unknown external push. In: Mechatronics and Automation, 2009. ICMA 2009. International Conference on, pp. 1928–1933 (2009). DOI 10.1109/ICMA.2009.5246563
43. Yeh, H., Curtis, S., Patil, S., van den Berg, J., Manocha, D., Lin, M.: Composite agents. In: Symposium on Computer Animation, pp. 39–47 (2008)
44. Yu, Q., Terzopoulos, D.: A decision network framework for the behavioral animation of virtual humans. In: Symposium on Computer animation, pp. 119–128 (2007)
45. Yu, W., Johansson, A.: Modeling crowd turbulence by many-particle simulations. Phys. Rev. E **76**, 046,105 (2007)
46. Zordan, V.B., Majkowska, A., Chiu, B., Fast, M.: Dynamic response for motion capture animation. ACM Trans. Graph. **24**(3), 697–701 (2005)