

Interactive Simulation of Local Interactions in Dense Crowds using Elliptical Agents

Sahil Narang^{1,¶,*}, Andrew Best^{1,¶}, Dinesh Manocha¹

1 Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA

¶These authors contributed equally to this work.

*** Corresponding Author**
Email: sahil@cs.unc.edu

Abstract

We present a practical approach for interactive crowd simulation based on elliptical agents. Our formulation uses a biomechanically accurate pedestrian representation to simulate different local interactions, including backpedaling, side-stepping, and shoulder-twisting. We present an efficient algorithm for local navigation and collision avoidance among multiple elliptical agents using velocity obstacles. Furthermore, we describe techniques to link the orientation of each elliptical agent to its velocity to automatically generate turning and lateral movements. In practice, our approach can simulate dense crowds of hundreds of pedestrians at interactive rates on a single CPU core. We highlight the performance in complex scenarios and validate our simulation results by comparing with real-world crowd videos and experiments.

1 Introduction

Pedestrian and crowd simulation has received considerable attention in different fields. Besides computer graphics, there is extensive work in biomechanics, psychology, robotics, and pedestrian dynamics. Biomechanics researchers evaluate various factors related to human locomotion; psychologists investigate the spatial and behavioral relationships of pedestrians and how those relationships affect their movement; robotics and vision researchers study various aspects of locomotion, including path planning, collision-avoidance, spatial flow, and visual appearance; the pedestrian dynamics community is interested in modeling the pedestrian flows for architectural design and evacuation planning. Many ideas from these fields are also used in computer graphics, including interactive crowd simulation for games and virtual reality, and are employed in generating special effects for movies and animation.

Some widely used crowd simulation algorithms are based on multi-agent models. In these simulations, each individual is represented as an independently sensing, planning, and moving *agent*. However, simulating the behavior and movement of crowds can be challenging. Many applications model large crowds consisting of hundreds or thousands (or more) agents. Simple, individual movements can result in emergent crowd behaviors, and it is important to simulate these collective patterns. Furthermore, many interactive applications must perform these computations in a tens of milliseconds or less. Given these challenges, most prior work in crowd simulation uses a simple representation for each agent as a 2D circular disc in a plane. There is extensive work on computing agent trajectory and behavior based on 2D disc representation using force-based [1], velocity-based [2, 3] rule-based [4], and cell-based [5] methods, among others. Most commercial and research systems for pedestrian and crowd simulation use such disc shapes and navigation algorithms.

Human pedestrians exhibit a large variety of behaviors and local interactions in tight spaces and dense situations, including movements and salient behaviors that are used to avoid other agents or obstacles and navigate through congestion and constriction [6, 7]. It is not uncommon for pedestrians to twist their shoulders and torsos to reduce their profile when passing one-another. They also perform sidestep movements or backpedal to avoid unexpected obstacles and make way for others. Current crowd simulation algorithms are unable to model many of these interactions using radially-symmetric discs. Furthermore, disc-based representations tend to overestimate the volume exclusion and restrict the maximum density in a crowd simulation.

Movements such as shoulder-turning, backpedaling, and side-stepping depend on the orientation of the agent. Some researchers impose additional restrictions with radially symmetric discs, including applying an orientation used only to model these complex behaviors or one that assumes that the agent is always facing along its velocity vector [8, 9]. However, these techniques tend to be non-intuitive and are not able to generate all

of the local interactions. Furthermore, rendering scenes with such disc-based agents can be difficult because artificial orientation constraints can result in inconsistent and erratic animation. Some of these rendering and foot-skating problems can be overcome using footstep planning [10], although most footstep models tend to be non-convex and dynamic.

Prior studies in pedestrian dynamics, psychology and biomechanics have argued that 2D ellipses are more appropriate approximations of a human body in terms of shape and movement [11–14]. Ellipses are orientable shapes and the orientation information can be used to model local interactions. However, the use of elliptical shapes for crowd simulation typically yields many navigational and computational challenges. Force-based methods may encounter radial forces as the closest point between two arbitrarily oriented ellipses is often not along the line connecting their centers, which can vary agent behaviors based on their orientation. Velocity obstacle based techniques must compute the Minkowski sums of two ellipses, which is considerably more expensive than the Minkowski sum of circles. Cell-based methods require computing complex decompositions of 2D planes into complementary and overlapping grids, while updating them for dynamic obstacles.

Main Results: We present an efficient agent-based crowd simulation algorithm for elliptical agents. Our approach includes two novel components. We have developed an efficient collision-free elliptical navigation algorithm based on velocity obstacles and an algorithm for automatically updating the orientation of each elliptical agent along the trajectory. We present techniques to link the orientation of each elliptical agent to its velocity and to compute side-stepping and shoulder turning movements based on empirical observations. Overall, our approach can simulate different local interactions among hundreds of elliptical agents at interactive rates on commodity CPU’s. Compared with prior methods, some of the novel components of our work include:

1. Simulating human-like local interactions and collision-avoidance behaviors, including side-stepping, shoulder-turning, and backpedaling.
2. A method to link the orientation computation to velocity, which reduces the simulation cost and dimensionality of motion planning.
3. Validation with pedestrian experiments including comparison to video data and behavior classification experiments.

Our overall formulation is simple. As compared to other geometric algorithms [15] for collision avoidance between elliptical agents, our overall algorithm for local collision avoidance and orientation computation results in *one order* of magnitude performance improvement. We demonstrate the effectiveness of our algorithm by testing it on multiple, dense scenarios. We also validate our method by reproducing the trajectories observed in real-world pedestrian videos and comparing emergent behaviors and interactions with captured pedestrian data.

The remainder of the paper is organized as follows. In Section 2, we provide a brief overview of prior work in crowd simulation and the use of orientable shapes. In Section 3, we introduce the notation and describe the elliptical representation and local collision-avoidance algorithm. In Section 4, we describe our approach to update the orientation of an elliptical agent. The overall crowd simulation algorithm and implementation details are described in Section 5. We highlight the performance of our algorithm and compare the results with real-world crowd videos in Section 6 and conclude in Section 7.

2 Related Work

In this section, we provide a brief overview of prior work in crowd simulation and use of elliptical shapes for modeling agents.

2.1 Crowd Simulation

There has been extensive work in simulating crowds or pedestrians in the fields of computer graphics, robotics and pedestrian dynamics for more than three decades. Some widely used approaches are based on cellular-automata [5], force-based computations [1, 8, 16], field-based methods [17, 18], geometric computations based on velocity obstacles [2], steering models [4], vision-based [19], continuum techniques [20–22], cognitive models and decision networks [23, 24], and data-driven simulation [3, 25, 26], among other approaches. These methods model different aspects of crowd simulation, including local collision avoidance, emergent behaviors, high-level behavior modeling, etc. However, almost all these methods model each agent as a 2D circle or disc on a plane to compute its trajectory or behavior. Most of these efficient local navigation and collision-avoidance methods exploit the fact that the agent can be conservatively approximated by a disc.

2.2 Lateral Motion and Orientable Shapes

Earlier work in pedestrian dynamics and biomechanics has argued for the use of elliptical shapes. Fruin [12] proposed the use of elliptical pedestrians for planning and design. G erin-Lajoie et al. [11] demonstrated the relationship between stride-length, density, and pedestrian movement shape to make a case for elliptical personal space. Templer [27] showed the presence of the sensory zone around a pedestrian using elliptical models. Imanishi and Sano [6] showed the importance of side-stepping and shoulder turning in avoidance behaviors from laboratory experiments. Singh et al. [28] present an algorithm for human-like steering in dynamic crowds based on a simple biomechanical footstep model and combine it with space-time planning to generate subtle navigation behaviors, including side-stepping.

There has been some research performed with respect to representing each agent as an orientable shape. Giese et al. [29] used the concept of reciprocal rotation to extend the reciprocal velocity obstacle (RVO) algorithm to polygons. Chraïbi et al. [14] applied social forces to deformable discs to simulate ellipse-like local behaviors. Johansson et al. [9] applied elliptical forces to disc-shaped agents. Karamouzas et al. [30] used velocity obstacles to navigate disc-based agents in deformable orientable formations. There is some work on modeling the lateral motion in crowd simulation. Hughes et al. [7] used pedestrian experiments to generate discrete side-stepping interruptions. Choi et al. [31] used precomputed motion patches to generate appropriate animation in complex environments. van Basten et al. [32] used interpolation techniques to generate parametrized footsteps for articulated characters. Our approach is based on explicit representation of 2D elliptical agents and differs from these methods.

3 Elliptical Agents

In this section, we introduce our notation and describe our elliptical representation along with the local collision avoidance algorithm.

3.1 Elliptical Representation

We assume that the underlying multi-agent simulation algorithm uses a high-level module that computes a new preferred velocity for each agent during each step of the simulation. This new preferred velocity could be computed based on a high-level behavior module or a global path planning algorithm. The preferred velocity is the velocity that optimally leads the agent toward its current goal if there were no other agents or obstacles in the scene. The current velocity is the actual velocity computed at the current time step to avoid collisions with the obstacles and other agents in the scene. Our agents are

represented in terms of their shape dimensions, current position, current velocity, and preferred velocity.

Notation: For an agent A , \vec{p}_A , \vec{v}_A , and \vec{v}_A^0 , denote the current position of the agent’s center of mass, current velocity, and preferred velocity. Each elliptical agent is represented using the following 10-dimensional state vector $[\vec{p}, \vec{v}, \vec{v}^0, \vec{o}, s^{maj}, s^{min}]$, where \vec{o}_A denotes the agent’s orientation. Following [33], we define orientation as “the angle of the vector perpendicular to the line connecting the shoulders and directed towards the front of the person”. s_A^{maj} and s_A^{min} denote the length of the agent’s semi-major and semi-minor axes, respectively (see Fig. 1). By contrast, a circular agent X is represented as a 7-dimensional vector $[\vec{p}_X, \vec{v}_X, \vec{v}_X^0, r_X]$, where r_X is its radius. In the rest of the paper, we use symbols A and B to denote elliptical agents and X and Y to denote disc-shaped agents.

It is understood that an elliptical approximation is more appropriate for representing human motion, particularly in cases of high-density or tight spaces, which is illustrated in Fig. 1; disc-based pedestrians overestimate the volume exclusion substantially more than a corresponding ellipse. Moreover, the use of discs restricts the maximum density and prevents the simulated crowds from reaching real measured densities.

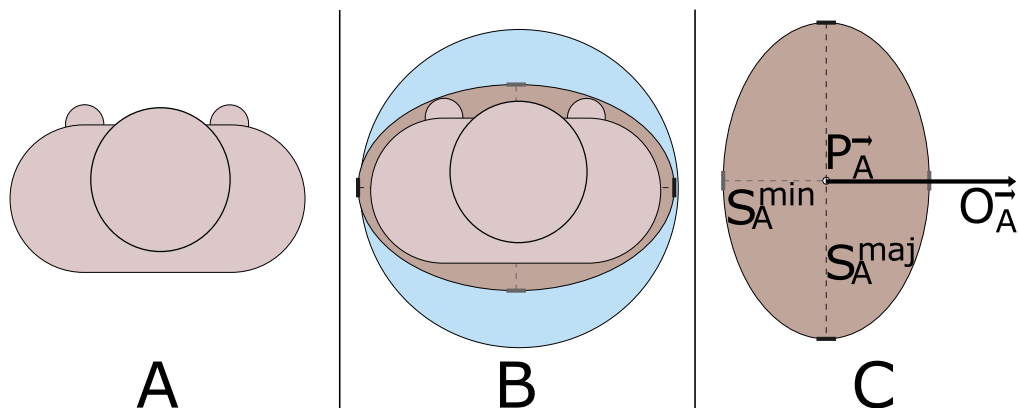


Figure 1. Approximating Pedestrians. (A). A sample human pedestrian of approximately average shoulder-width and body depth, 0.228 m and 0.149 m, respectively. (B) The typical disc-based approximation (blue) of the human overestimates its depth by over 50%. Our proposed elliptical approximation (brown) captures the pedestrian shape accurately and can keep track of the orientation. (C) Representation: Each agent A ’s state vector consists of position \vec{p}_A , orientation \vec{o}_A , current velocity \vec{v}_A , preferred velocity \vec{v}_A^0 , semi-major axis s_A^{maj} and semi-minor axis s_A^{min} .

Given the preferred velocity, the underlying approach uses a local collision-avoidance technique to compute a new velocity, \vec{v}^{new} , and orientation, \vec{o}^{new} , for each elliptical agent. Furthermore, each agent has access to the directly observable properties of its neighbors to compute its current velocity. These properties include \vec{p} , \vec{v} , \vec{o} , s^{maj} and s^{min} . However, properties such as \vec{v}^0 are considered internal or intrinsic to the agent and cannot be observed by other agents. We assume that all the agents in the simulation are using the same local collision-avoidance and navigation strategies. To ensure consistency, the current velocity \vec{v} , is set to the new velocity \vec{v}^{new} for each agent simultaneously at the end of the simulation step. Likewise, the current orientation \vec{o} , is set to the new orientation \vec{o}^{new} .

3.2 Local Collision Avoidance

Our local collision avoidance algorithm is based on velocity obstacles that have been used for collision-avoidance purposes in robotics and crowd simulation [2, 34]. However, all prior velocity obstacle-based algorithms are restricted to circular agents. We outline our efficient and conservative formulation for elliptical agents which computes a collision-free velocity for each agent and provide full details in [35].

Velocity Obstacles: For two agents A and B , centered at \vec{p}_A and \vec{p}_B , respectively, the velocity obstacle of A induced by B , denoted by $VO_{A|B}^\tau$, constitutes the set of velocities for A that would result in a collision with B at some time before τ . By definition, agents A and B are guaranteed to be collision-free for at least time τ , if $\vec{v}_A - \vec{v}_B \notin VO_{A|B}^\tau$ [34]. In this case, $\vec{v}_A - \vec{v}_B$ is the relative velocity of A and B . We use a geometric interpretation of this condition. In general, let V denote the set of all available velocities for an agent. At each simulation step, the agent must choose a velocity $\vec{v}^{new} \in V$ s.t. \vec{v}^{new} lies outside the velocity obstacles defined by all the neighboring agents and obstacles. This is a sufficient condition for collision-free navigation for at least time τ . We refer the reader to [35] for details on computing the velocity obstacle for neighboring elliptical agents.

3.2.1 Computing Neighboring Agent Constraints

We use the velocity obstacle to compute the set of permitted velocities for an agent. Given the velocity obstacle $VO_{A|B}^\tau$, we construct the set of permitted velocities for A for optimal reciprocal collision avoidance [2]. Each agent determines the set of velocities excluded by $VO_{A|B}^\tau$ and the minimum deviation from the agents' relative velocity to guarantee collision avoidance. As described in Section 3.1, all agents use the same collision-avoidance strategy. Therefore, agent A is responsible for adapting its velocity by *half* the required change assuming that B will do the same. The constraint takes the form of a half-plane of excluded velocities. One constraint half-plane is constructed for each neighboring agent and obstacle. After all the constraints have been created, a collision-free velocity is chosen from the set of remaining permissible velocities.

3.2.2 Computing Neighboring Obstacle Constraints

Without loss of generality, we can assume that all obstacles in the scene are triangulated and their projections on the 2D plane are given as a collection of line segments. We follow the same basic approach described in Section 3.2.1 with the exception that in this case, the agent must take full responsibility to avoid a collision with the static obstacle. An agent A will collide with obstacle O within the time τ if its velocity \vec{v}_A is inside $VO_{A|O}^\tau$.

3.2.3 Choosing a Collision-free velocity

At every simulation step, we construct the half-plane constraints for each neighboring agent and obstacle. The set of permitted velocities for agent A is simply the convex region $EORCA_A^\tau$, given by the intersection of the half-planes of the permitted velocities induced by all the neighboring agents and obstacles.

$$EORCA_A^\tau = \bigcap_{B \neq A} EORCA_{A|B}^\tau \quad (1)$$

The agent is responsible for selecting a new velocity v_A^{new} from $EORCA_A^\tau$ that minimizes the deviation from its preferred velocity v_A^{pref} .

$$v_A^{new} = \min_{v \in EORCA_A^\tau} \|v - v_A^{pref}\| \quad (2)$$

Eq. 1 and 2 can be solved efficiently with an expected runtime of $O(n)$ using linear programming, where n is the total number of constraints. With elliptical agents, the shape of the velocity obstacle, the tangents, and the nearest point operations are governed by the agent’s orientation and by the orientation of the obstacles. Therefore, we require efficient acceleration techniques to perform these computations. We refer the reader to [35] for more details.

4 Orientation Computation

In general, pedestrians face – or are oriented toward – their direction of motion. This fact is used by current crowd simulation algorithms that typically extrapolate the orientation from an agent’s current velocity. However, pedestrians often twist their shoulders and/or take sidesteps to maneuver through dense or tight spaces. They often backpedal momentarily to give way to oncoming traffic. In these cases, their orientation is no longer in line with the direction of their movement. These local behaviors are not modeled by current simulation algorithms because, among other reasons, the commonly used disc representation inherently lacks the capability to model orientation. Some techniques augment the disc with an orientation vector, which is used for animation and rendering [8,9], but don’t offer sufficient capability to model these local interactions.

We present a novel orientation computation algorithm for elliptical agents that accounts for shoulder turning, lateral motion, and backpedaling. Due to their inherent asymmetry, elliptical agents can change their orientation to decrease the portion of body width in the walking direction, allowing them to navigate through small gaps and generate more human-like trajectories. This implies that we now need to explicitly track the orientation of an agent. Furthermore, it expands the underlying configuration space of each agent to three dimensions – $[x, y, \theta]$. We decouple the problem into computing the current velocity based on the current positions of the ellipse (as described in Section 3.2) and separately computing the orientation based on the current velocity and current orientation.

4.1 Empirical Observations

Imanishi and Sato [6] investigated pedestrian avoidance behaviors in crossing flows under laboratory conditions. They conducted experiments in which a pedestrian, the “traverser”, crosses a crowd of pedestrians, referred to as “pedestrians in flow” (see the figure in the appendix). Each experiment was parametrized based on the density of pedestrians in a flow and the angle at which the *traverser* approaches the crowd flow. They observed that pedestrians avoided collisions by adjusting their walking speed, walking route (detouring), and shoulder turning. Moreover, Imanishi and Sato quantitatively classified each of these behaviors into *four levels of avoidance*: no, potential, weak and strong avoidance. They observed that pedestrians did not use shoulder turning when there was enough margin space around them. For example, with a density of two *pedestrians/m²* and crossing angle of 180, pedestrians slightly reduced their speeds and avoided each other primarily by controlling their shoulder angles. They observed shoulder turns as high as 80 degrees. However, in situations with a crossing angle of 45 degrees with the same density, pedestrians mostly relied on speed reduction and only infrequently performed shoulder turns. Their results also demonstrate that higher pedestrian densities result in strong avoidance. Hughes et al. [7] also conducted experiments to observe the conditions under which humans exhibit lateral motion and concluded that lateral motion occurs when a collision is imminent and is mostly in tight spaces in which the pedestrian may not be able to navigate with a non-lateral orientation. Our approach is driven by these empirical observations.

4.2 Shoulder Turning

Given its new velocity, each agent evaluates the scalar space, termed as ‘‘clearance’’, available to it at a sequence of points along the line joining its current position and its estimated position in the next second ($\vec{p} + \frac{\vec{v}}{\|\vec{v}\|}$). Essentially, the line denotes the loci of the center of the ellipse during the interval $[t, t + 1]$. In general, the agent is expected to continue along with its current orientation, effectively translating along this line. Figure 2 describes the clearance computation at a given point w.r.t. the agent. The estimated clearance, c , available to the agent w.r.t. its current orientation is set to be the minimum clearance at the sampled points. Once computed, the algorithm chooses a new orientation o^{new} as given by the optimization:

$$\begin{aligned} & \underset{o_{new}}{\text{minimize}} && \|o_{new} - o\| \\ & \text{subject to} && w(o_{new}) \leq c, \end{aligned}$$

where w defines the cross-section width of the new orientation perpendicular to its new velocity. While computing the clearance, the agent only considers nearby agents and obstacles that are ahead with respect to its new velocity and are also visible to the agent at its current position.

Computing Clearance: Let \vec{q}_i denote the point at which we wish to compute the clearance. We define \hat{n}_i to be a unit vector perpendicular to the new velocity \vec{v}_i^{new} . Given a nearby agent j , we compute the nearest point \vec{p}_{ij} on j with respect to \vec{q}_i . Let c_{ij}^+ and c_{ij}^- denote the clearance available for agent i at point \vec{q}_i in the direction \hat{n}_i and $-\hat{n}_i$ respectively, with respect to agent j , computed as:

$$c = \|\vec{p}_{ij} - \vec{q}_i\| \quad (3)$$

$$a = (\vec{p}_{ij} - \vec{q}_i) \cdot \hat{n}_i \quad (4)$$

$$c_{ij}^+ = \min(c, c_{ij}^+) \iff a \geq 0 \quad (5)$$

$$c_{ij}^- = \min(c, c_{ij}^-) \iff a < 0, \quad (6)$$

where c_{ij}^+ and c_{ij}^- are initialized to infinity. Clearance c_{ij} at point \vec{q}_i can then be estimated as $c_{ij} = c_{ij}^+ + c_{ij}^-$ (Figure 2). Without loss of generality, the same equations can be applied to compute clearance with respect to a nearby obstacle j , where \vec{p}_{ij} represents the closest point on the obstacle to \vec{q}_i . Doing so for each nearby agent and obstacle, we can estimate the scalar space available to an agent at the query point \vec{q}_i .

To determine whether the new orientation is feasible, each agent projects their own position and the positions of their neighbors one step forward in simulation, assuming agent i adapts its new velocity \vec{v}_i^{new} and new orientation \vec{o}_i^{new} while other agents continue with their current velocity and orientation. We then explicitly check for collisions. If there is a collision, the orientation update is discarded and the agent maintains its prior orientation. The detailed algorithm is described in Table 1.

4.3 Lateral Motion and backpedaling

Agent i estimates the time to collision with respect to nearby agents and obstacles based on its new velocity v_i^{new} . It also uses the new velocity to compute the angle of deviation from the preferred direction of travel. If the deviation from the preferred direction is greater than threshold θ_{thresh} and there is an imminent collision – i.e., the time to collision is less than threshold ttc_{thresh} – the resulting algorithm performs lateral movement or backpedaling i.e. the agent adapts its new velocity without changing its orientation. We base our algorithm, described in Table 2, on the observations of Hughes et al. [7].

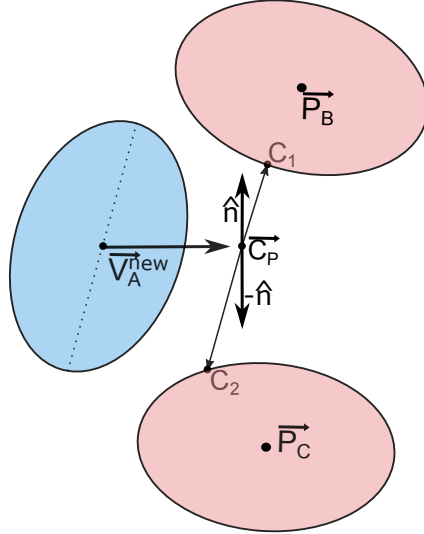


Figure 2. Computing Clearance. We compute the clearance at a point $\vec{C_P}$ that is nearby (e.g., 1 meter ahead) in the direction of the new velocity \vec{v}_A^{new} . \vec{c}_1 denotes a vector from $\vec{C_P}$ to the closest point on the nearest agent such that $\vec{c}_1 \cdot \hat{n} \geq 0$ where \hat{n} is a unit vector perpendicular to \vec{v}_A^{new} . Similarly, \vec{c}_2 denotes a vector from $\vec{C_P}$ to the closest point on the nearest agent such that $\vec{c}_2 \cdot \hat{n} < 0$. Clearance at $\vec{C_P}$ is set to $\|\vec{c}_1\| + \|\vec{c}_2\|$.

5 Interactive Crowd Simulation

In this section, we describe our overall crowd simulation algorithm. We briefly present acceleration structures that are precomputed to accelerate the runtime computation and also present details regarding implementation.

5.1 Collision avoidance in high density scenarios

Our collision-avoidance algorithm described in Section 3.2 is conservative. It is possible that in densely packed conditions, the set of permitted velocities may be an empty set, i.e., $EORCA_A = \emptyset$. In that case, linear programming will report that the solution set is infeasible. One way to address this scenario is to relax the constraints uniformly until there is a feasible velocity. This computation is performed by reducing to a three-dimensional linear programming where the third dimension is the signed distance to that specific constraint. Thus, we choose a velocity that minimally penetrates the constraints induced by other agents. The 3-D linear program is always feasible, and we can compute an optimal solution to this formulation. Because we have relaxed certain of the EORCA constraints, we can no longer guarantee that the new velocity would result in collision-free motion.

5.2 Overall Algorithm

Table 3 presents the overall simulation update mechanism for each agent. We first compute the EORCA half-plane constraints for nearby agents and obstacles, as described in Section 3.2. We use linear programming to compute the new velocity for the next time step (or the relaxation described above). After computing the new velocity, the algorithm determines whether the agent must side-step (Section 4.3), turn its shoulder (Section 4.2), or align its orientation with the new velocity. In any case, we use a threshold for the maximum angular acceleration α_{thresh} to bound the change in orientation. In order

Table 1. EvalShoulderTurn: Compute the preferred orientation for shoulder turning (if needed) for agent i .

Input: Position \vec{p}_i , new velocity \vec{v}_i^{new} , orientation \vec{o}_i , and simulation timestep Δt
Output: Boolean indicator for shoulder turn, $result$, and the preferred orientation \vec{o}_i^{pref}

```

result = false
 $\vec{C}P \leftarrow \vec{p}_i + \frac{\vec{v}_i^{new}}{\|\vec{v}_i^{new}\|}$ 
obs  $\leftarrow$  GetNeighboringObstacles( $\vec{p}_i$ )
agts  $\leftarrow$  GetNeighboringAgents( $\vec{p}_i$ )
ci  $\leftarrow$  getClearanceAtPoint( $\vec{C}P_i$ , obs, agts)
 $\vec{o}_i^{pref} \leftarrow$  getEllipseForClearance( $\vec{o}_i$ , ci)
if ( $\vec{o}_i^{pref} \neq \vec{o}_i$ ) then
    collision  $\leftarrow$  CollisionCheck( $\vec{p}_i$ ,  $\vec{v}_i^{new}$ ,  $\vec{o}_i^{pref}$ , agts, obs,  $\Delta t$ )
    if (collision  $\neq$  true) then
        result = true
    end if
end if

```

to maintain consistency, we store the new orientation and new velocity for each agent and perform updates at the end of the simulation step, as described by the following equations:

$$\begin{aligned} \vec{v}_i &\leftarrow \vec{v}_i^{new}, \\ \vec{o}_i &\leftarrow \vec{o}_i^{new}, \\ \vec{p}_i &\leftarrow \vec{p}_i + \Delta t * \vec{v}_i. \end{aligned}$$

5.3 Implementation

To perform interactive simulation with a large number of agents, we precompute an evenly spaced set of approximated ellipses in the angular range 0 to 2π with a given angular resolution. We also precompute the Minkowski Sum for every pair of discrete ellipses and some additional information during an offline phase. This data is stored on disk and can be accessed during runtime to speed up computations. More details on the precomputed data and the run time costs can be found in [35].

We implemented our algorithm in C++ on a windows 7 desktop PC. Timing results were generated on an Intel i7-4790 pc with 16GB of ram. All comparison algorithms – ORCA [2], social forces [1], and our EORCA algorithm – were parallelized per agent on 4 cores. Table 4 shows the per-agent parameters used to generate the results in our simulation.

6 Performance and Validation

In this section, we highlight the performance of our algorithm on different benchmarks and validate the results with some captured real-world crowd videos and trajectories.

6.1 Validation

We validate the performance of our algorithm by reproducing the live pedestrian experiments reported by [6] and [36] that highlight pedestrian interaction in crossing flows and bidirectional flows respectively. We further highlight the performance of our algorithm

Table 2. EvalLateralMotion(): Determine if agent i should execute lateral motion and backpedal

Input: Position \vec{p}_i , new velocity \vec{v}_i^{new} , preferred velocity \vec{v}_i^0 , threshold from deviation from preferred direction θ_{thresh} and threshold for time to collision ttc_{thresh}

Output: Boolean indicator for sidestepping $result$.

```

 $ttc \leftarrow \infty$ 
 $result = false$ 
for each neighboring agent  $j$  do
   $\hat{k} \leftarrow \frac{\vec{p}_j - \vec{p}_i}{\|\vec{p}_j - \vec{p}_i\|}$ 
   $\vec{v}_{ij}^{tang} \leftarrow ((\vec{v}_i^{new} - \vec{v}_i) * \hat{k}) * \hat{k}$ 
   $ttc_j \leftarrow \frac{\|\vec{p}_j - \vec{p}_i\|}{\|\vec{v}_{ij}^{tang}\|}$ 
  if  $ttc_j \leq ttc$  then
     $ttc \leftarrow ttc_j$ 
  end if
end for
for each neighboring obstacle  $obs$  do
   $\vec{p}_j \leftarrow \text{ClosestPtOnObstacle}(\vec{p}_i, obs)$ 
   $\hat{k} \leftarrow \frac{\vec{p}_j - \vec{p}_i}{\|\vec{p}_j - \vec{p}_i\|}$ 
   $\vec{v}_j^{tang} \leftarrow ((\vec{v}_i^{new} - \vec{v}_i) * \hat{k}) * \hat{k}$ 
   $ttc_j \leftarrow \frac{\|\vec{p}_j - \vec{p}_i\|}{\|\vec{v}_j^{tang}\|}$ 
  if  $ttc_j \leq ttc$  then
     $ttc \leftarrow ttc_j$ 
  end if
end for
 $\theta_{dev} \leftarrow \text{acos}(\frac{\vec{v}_i^{new} * \vec{v}_i^0}{\|\vec{v}_i^{new}\| * \|\vec{v}_i^0\|})$ 
if  $\text{abs}(\theta_{dev_i}) \geq \theta_{thresh}$  and  $ttc \leq ttc_{thresh}$  then
   $result = true$ 
end if

```

on a set of illustrative demonstrations based on real-world scenarios in which lateral motion and high-density situations are common.

6.1.1 Crossing Flows

Recently, Imanishi et. al. [6] sought to study pedestrian avoidance behaviors in crossing flows. They devised an experiment in which a group of 16 pedestrians with density $\rho \in \{0.25, 1, 2\}p/m^2$ move through an open room. A single person crosses the group at an angle $\theta \in \{45, 90, 135, 180\}$ deg. Three trials were conducted for each combination of density and crossing angle. The authors measured the level of occurrence of each of three local collision-avoidance behaviors: speed reduction, detouring, and shoulder-twisting. For each combination of density and crossing angle, the authors further classify the degree of collision avoidance via shoulder turns into one of four categories: (S)Strong avoidance (36 – 90 deg), (W)Weak avoidance (24 – 36 deg), (P)Potential avoidance (12 – 24 deg) and (N)No avoidance (≤ 12 deg).

To validate our algorithm, we faithfully reproduced their experiments. Pedestrians were modelled as elliptical agents with average human shoulder-width and body-depth and a preferred speed of 1.3m/sec. For each combination of density, ρ and crossing angle θ , we recorded the maximum shoulder turn made by the traverser. We classify the shoulder turns in accordance with the classification presented by the authors. We report these observations in Table 5 along with the observations made by the authors. For our

Table 3. Simulation Update for agent i

Input: State Vector $[\vec{p}_i, \vec{v}_i, \vec{v}_i^0, \vec{o}_i, s_i^{maj}, s_i^{min}]$ for agent i , simulation time step Δt , threshold for deviation from preferred direction θ_{thresh} , threshold for time to collision ttc_{thresh} , threshold for angular acceleration α_{thresh} , and threshold for lateral motion time $lateral_{thresh}$.

Output:

```

 $agtLines_i \leftarrow \text{ComputeAgentConstraints}()$ 
 $obsLines_i \leftarrow \text{ComputeObsConstraints}()$ 
 $\vec{v}_i^{new} \leftarrow \text{SolveLinearProgram}(agtLines_i, obsLines_i)$ 
 $\vec{o}_i^{new} \leftarrow \frac{\vec{v}_i^{new}}{\|\vec{v}_i^{new}\|}$ 
 $lateral \leftarrow \text{EvalLateralMotion}(\vec{p}_i, \vec{v}_i^{new}, \vec{v}_i^0, \theta_{thresh}, ttc_{thresh})$ 
if ( $lateralSeq \leq lateral_{thresh}$  &&  $lateral == true$ ) then
   $lateralSeq = lateralSeq + \Delta t$ 
   $\vec{o}_i^{new} \leftarrow \vec{o}_i$ 
else
   $lateralSeq = 0$ 
  ( $turn, \vec{o}^{pref}$ )  $\leftarrow \text{EvalShoulderTurn}(\vec{p}_i, \vec{v}_i^{new}, \vec{o}_i, \Delta t)$ 
  if ( $turn == true$ ) then
     $\vec{o}_i^{new} \leftarrow \vec{o}^{pref}$ 
  end if
end if
if ( $\vec{o}_i^{new} \neq \vec{o}_i$ ) then
   $\vec{o}_i^{new} \leftarrow \text{ClampAngularAcc}(\vec{o}_i, \vec{o}_i^{new}, \alpha_{thresh}, \Delta t)$ 
end if

```

s^{maj}	s^{min}	m	τ	$\tau_{obstacle}$	α_{thresh}	θ_{thresh}	ttc_{thresh}	$lateral_{thresh}$
0.2286	0.149	100	1.0	1.0	360	10	6	3

Table 4. Simulation Parameters. The parameters used in our experiments. In order: semi-major axis length(m), semi-minor axis-length(m), number of samples, time horizon for agent-agent collision avoidance(sec), time horizon for agent-obstacle collision avoidance(sec), maximum angular acceleration (deg/sec), threshold for deviation from preferred direction (deg), threshold for time to collision(sec) and threshold for lateral motion time(sec).

model, we also report the maximum degree of shoulder turn made by the traverser.

Our results of our simulation are summarized as follows:

- Consistent with real world observations, shoulder turning was negligible in cases where the traverser had enough margin of space to travel with its current orientation ($\rho \in \{0.25, 1\}$ $people/m^2$, $\theta \in \{45, 135\}$ deg).
- The degree of shoulder turning increased as the available space decreased in higher densities ($\rho \in \{1, 2\}$ $people/m^2$, $\theta = 90$ deg) as was observed by Imanishi et. al.
- Our model was able to match the observed level of shoulder turn in every case except when the traverser attempts to cross the crowd of density 2 $people/m^2$. In this case, the velocity constraints caused the traverser to detour around the crowd flow thus, allowing him the clearance to continue without shoulder turning. However, in the study, the traverser was instructed to “walk straight towards the destination”, thus walking through the crowd which required him/her to shoulder turn.

Density (<i>people/m</i> ²)	Crossing Angle							
	180		135		90		45	
	IS14	Ours[max]	IS14	Ours[max]	IS14	Ours[max]	IS14	Ours[max]
0.25	-	W[3.4]	-	W[2.1]	-	W[3.75]	-	W[2.91]
1	-	W(1.626)	W	W[25.1]	S	S[90]	W	W[30.01]
2	S	N[0.12]	S	S[45.5]	S	S[83]	W	W[29.7]

Table 5. Validation results for crossing flow. We validate our algorithm with the study conducted by [6] (shown as IS14). For each combination of density and crossing angle, the authors classify the degree of collision avoidance via shoulder turns into one of four categories: (S)Strong avoidance (36 – 90 deg), (W)Weak avoidance (24 – 36 deg), (P)Potential avoidance (12 – 24 deg) and (N)No avoidance (≤ 12 deg). The table presents the classification of the observed degree of collision avoidance shown by the pedestrian crossing the flow (IS14) as well as the one simulated by our algorithm (shown as Ours). We also provide the maximum angle (max) of shoulder turn observed in our simulation. Our algorithm based on elliptical agents generates results that are consistent with human behavior.

6.1.2 Bidirectional Flow

We also validated our model by reproducing the experiments of [36]. This experiment tracked two groups of pedestrians moving in opposing directions through a hallway. The researchers varied the initial density of the pedestrians and observed the relationship between speed and density in the hallway. Behaviors such as shoulder turning and sidestepping can be frequently seen in cases of high densities ($2 - 2.5 \text{ people/m}^2$). We reproduced the experiment and ran our simulation using elliptical agents with initial density $\rho \in \{0.5, 1.0, 1.5, 2.0, 2.5\} \text{ people/m}^2$. We observed emergent behaviors such as lane formation for densities as high as 2 people/m^2 . This implied that most agents could smoothly navigate through the hallway without the need for shoulder turns and sidesteps. However, at a higher density of 2.5 people/m^2 , shoulder turning and sidestepping were very frequent, in accordance with the captured data. We generated a visual rendering of the high-density experiment ($\rho = 2 \text{ people/m}^2$) and highlighted the occurrence of shoulder-twisting and side-stepping behaviors using our algorithm (Figure 3).

6.2 Benchmarks

We devised a set of benchmark scenarios to highlight how elliptical pedestrians can be used to generate real-world crowd behaviors. These demonstrations clearly illustrate frequently seen avoidance behaviors such as shoulder turns, lateral motion and back pedalling.

Aircraft Unloading: Our first benchmark shows the agents exiting a commercial aircraft (Fig. 4(A)(B)(C)). The agents stand up, side-step to the aisle, fetch their luggage, and exit the aircraft. The seat clearance is not large enough for agents to turn and walk to the aisle, thus forcing the elliptical agents to sidestep to the aisle. Furthermore, the aisles are not wide enough for two disc-based agents to pass each other. However, two elliptical agents can pass each other by turning their shoulders and sidestepping.

Subway Station: In this scenario, depicted in Fig. 4 parts (D),(E) and (F), a group of agents rush the platform and wait for the train. When the train arrives, these agents attempt to board the train while those on the train attempt to exit. This creates a bottleneck and is an ideal scenario for the target behaviours. Using our models, agents backpedal and shoulder-twist to allow those agents disembarking to pass before entering the subway car. The elliptical agents are able to make room for the exiting passengers with limited motion and without unnecessary rotations, resulting in natural looking motion.

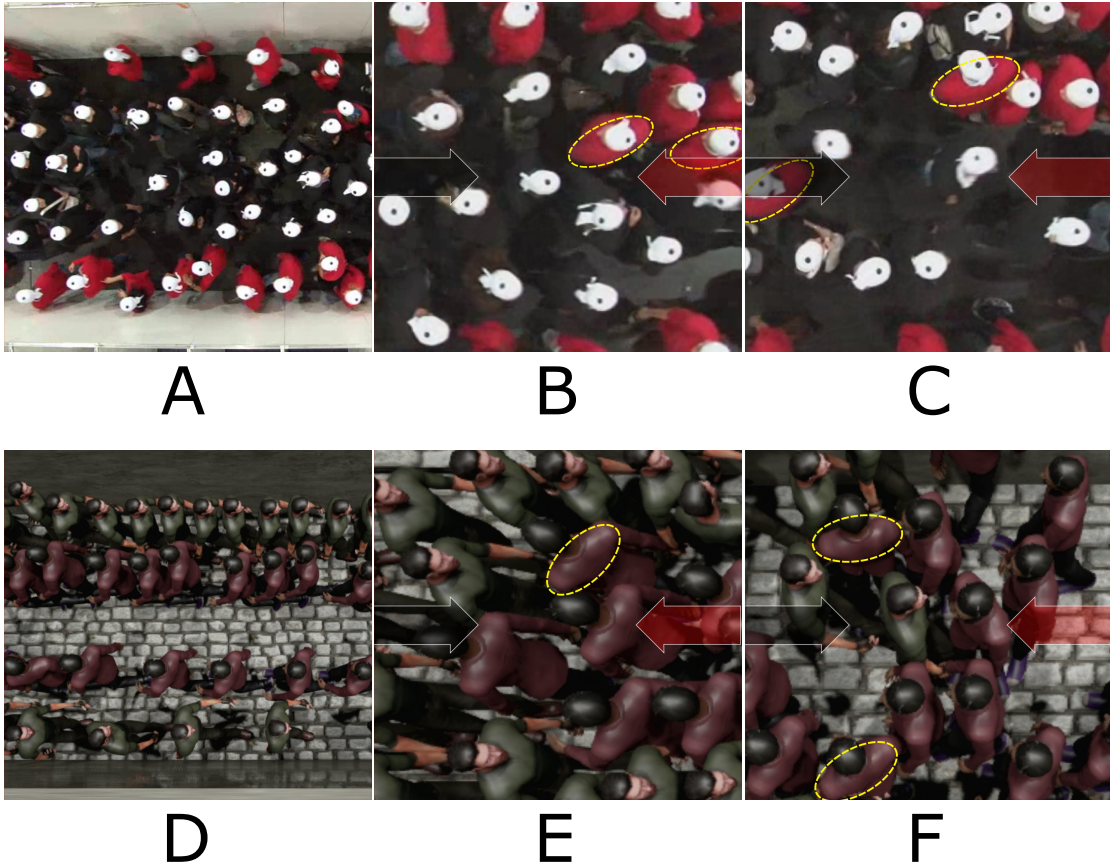


Figure 3. Validation results for bidirectional flow. (Top) Frames from captured crowd footage [36]. Shoulder twisting is a commonly observed phenomenon in high-density crowds as shown in these still frames. In each frame, several twisting agents have been highlighted in yellow with ellipses. (Bottom) We use elliptical agents to generate such local behaviors automatically. The agents shoulder turn and sidestep (E and F) to make their way through regions of high densities.

6.3 Computation Time

We include the average frame computation time for crowd simulation with elliptical agents and compare that with a crowd simulation with disc-based agents. Table 6 shows the running time of each scenario for our elliptical agents vs. disc-based agents using ORCA [2]. The use of precomputed data structures described by [35] improves the runtime performance of EORCA algorithm by almost 40 times. As mentioned earlier, exact geometric computations using ellipses can be very expensive operations for interactive simulation of large number of agents. The combination of approximations, setting up EORCA constraints, linking orientation to the velocity, and the precomputed data structures improves the overall performance by one order of magnitude.

7 Limitations, Conclusions, and Future Work

We have developed an efficient collision-free navigation algorithm for elliptical agents and compute their orientation based on real-world observations. In particular, we present algorithms that link the orientation of each agent to its velocity and compute appropriate side-stepping and turning motions. We have validated the results by comparing the local



Figure 4. Benchmarks. (Top) Pedestrians exit a commercial aircraft. Our algorithm is able to capture the lateral steps individuals take when exiting from their row and the shoulder-twisting that occurs as they pass one another in the aisle. (Bottom) Pedestrians enter a subway station and wait for the train to arrive. When the doors open, the waiting pedestrians step back to let those aboard exit before entering the subway train. Agents sidestep, backpedal (F) and shoulder turn (E) as they make their way into and out of the train.

interactions with real-world behaviors. Overall, we have presented a practical algorithm for multi-agent crowd simulation with elliptical agents.

Our approach has some *limitations*. The collision-avoidance and navigation algorithms that are based on elliptical agents are more expensive (e.g., one order of magnitude) and more complex than discs. As a result, their utility may be limited to dense situations in which such local interactions are important. The collision-avoidance algorithm tends to be conservative. Moreover, the orientation computation is only able to model some – and not all – of the local interactions related to turning, lateral motion and backpedaling. The overall approach is also limited to crowd simulation algorithms that reduce trajectory and behavior computation to the preferred velocity specification at each frame.

There are many avenues for future work. In addition to overcoming these limitations, we would like to test our algorithm in different settings and evaluate its performance. It would be useful to develop techniques for flocking or well-known behaviors for elliptical agents (similar to [4]) and also combine these with high-level behavior modeling and footstep planning [28]. Moreover, we would like to further improve the runtime performance using parallelization techniques.

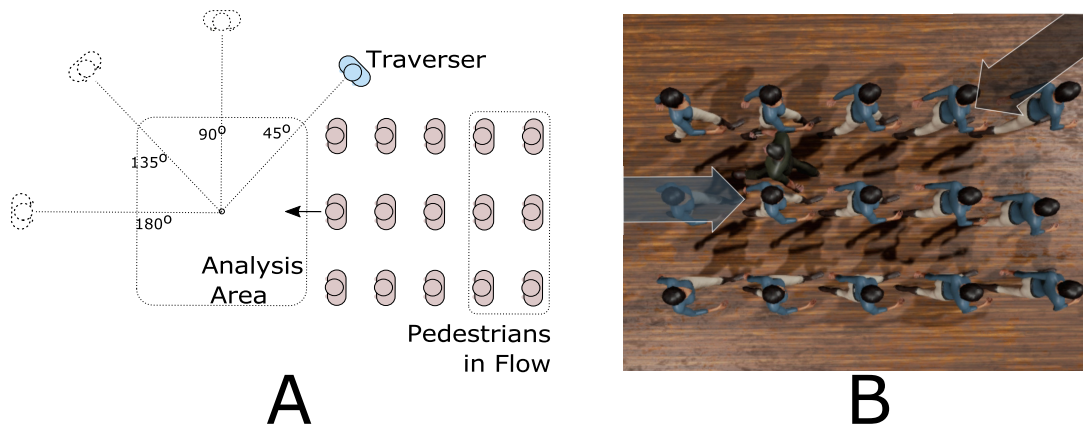
Scene	Num. agents	Disc-based Agents	Elliptical Agents
Bidirectional Flow	200	1.70	5.40
Subway	183	1.98	9.08
Crossing	16	3.18	12.68
Aircraft	157	1.13	8.43

Table 6. Average frame computation time (ms). We find that simulating elliptical agents using our algorithm is one order of magnitude more expensive than disc-based agents.

References

1. Helbing D, Molnár P. Social force model for pedestrian dynamics. *Physical Review E*. 1995 May;51(5):4282–4286.
2. van den Berg J, Guy SJ, Lin M, Manocha D. Reciprocal n-body Collision Avoidance. In: *Inter. Symp. on Robotics Research*; 2011. p. 3–19.
3. Pettré J, Ondřej J, Olivier AH, Cretual A, Donikian S. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In: *Symposium on Computer Animation*; 2009. p. 189–198.
4. Reynolds CW. Flocks, herds and schools: A distributed behavioral model. In: *Proc. of SIGGRAPH*; 1987. .
5. Schadschneider A. Cellular Automaton Approach to Pedestrian Dynamics - Theory. *Pedestrian and Evacuation Dynamics*. 2002;p. 75–86.
6. Imanishi M, Sano T. Level of Avoidance in Crossing Pedestrian Flow. *Transportation Research Proceedings*. 2014;2(0):367 – 375. Available from: <http://www.sciencedirect.com/science/article/pii/S2352146514000702>.
7. Hughes R, Ondrej J, Dingliana J. Holonomic Collision Avoidance for Virtual Crowds. In: Koltun V, Sifakis E, editors. *Symposium on Computer Animation*. Eurographics Association; 2014. p. 103–111. Available from: <http://dx.doi.org/10.2312/sca.20141128>.
8. Karamouzas I, Heil P, van Beek P, Overmars MH. A Predictive Collision Avoidance Model for Pedestrian Simulation. In: *Motion in Games*. vol. 5884 of *Lecture Notes in Computer Science*. Springer; 2009. p. 41–52.
9. Johansson A, D H, Shukla PK. Specification of the social force pedestrian model by evolutionary adjustment to video tracking data. *Advances in Complex Systems*. 2007;10:271–288.
10. Berseth G, Kapadia M, Faloutsos P. Robust Space-Time Footsteps for Agent-Based Steering. *Computer Animation and Virtual Worlds*. 2015;.
11. Gérin-Lajoie M, Richards CL, McFadyen BJ. The Negotiation of Stationary and Moving Obstructions During Walking: Anticipatory Locomotor Adaptations and Preservation of Personal Space. *Motor Control*. 2005;9:242–269.
12. Fruin JJ. Pedestrian planning and design. *Metropolitan Association of Urban Designers and Environmental Planners*; 1971. Available from: <http://books.google.com/books?id=AydSAAAAMAAJ>.
13. Pauls JL. Suggestions on evacuation models and research questions. In: *Conference Proceedings of the 3rd International Symposium on Human Behaviour in Fire*; 2004. .
14. Chraïbi M, Seyfried A, Schadschneider A. Generalized centrifugal-force model for pedestrian dynamics. *Phys Rev E*. 2010;82(4):046111.
15. Lee IK, Kim MS, Elber G. Polynomial/Rational Approximation of Minkowski Sum Boundary Curves. *Graphical Models and Image Processing*. 1998;60(2):136 – 165. Available from: <http://www.sciencedirect.com/science/article/pii/S1077316998904646>.

16. Pelechano N, Allbeck J, Badler N. Controlling individual agents in high-density crowd simulation. In: Symposium on Computer Animation; 2007. p. 99–108.
17. Patil S, van den Berg J, Curtis S, Lin MC, Manocha D. Directing crowd simulations using navigation fields. *IEEE TVCG*. 2010;p. 244–254.
18. Sung M, Gleicher M, Chenney S. Scalable behaviors for crowd simulation. *Computer Graphics Forum*. 2004;23(3):519–528. Available from: <http://dx.doi.org/10.1111/j.1467-8659.2004.00783.x>.
19. Ondřej J, Pettré J, Olivier AH, Donikian S. A synthetic-vision based steering approach for crowd simulation. In: *Proc. of ACM SIGGRAPH*; 2010. p. 123:1–123:9.
20. Treuille A, Cooper S, Popović Z. Continuum crowds. In: *Proc. of ACM SIGGRAPH*; 2006. p. 1160–1168.
21. Narain R, Golas A, Curtis S, Lin MC. Aggregate dynamics for dense crowd simulation. *ACM Trans Graph*. 2009;28:122:1–122:8.
22. Kapadia M, Singh S, Hewlett W, Faloutsos P. Egocentric affordance fields in pedestrian steering. In: *Proceedings of the Symposium on Interactive 3D Graphics and Games*; 2009. p. 215–223.
23. Funge J, Tu X, Terzopoulos D. Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In: *Proc. of SIGGRAPH*; 1999. p. 29–38.
24. Yu Q, Terzopoulos D. A decision network framework for the behavioral animation of virtual humans. In: *Symposium on Computer animation*; 2007. p. 119–128.
25. Lee KH, Choi MG, Hong Q, Lee J. Group behavior from video: a data-driven approach to crowd simulation. In: *Symposium on Computer Animation*; 2007. p. 109–118.
26. Lerner A, Chrysanthou Y, Lischinski D. Crowds by Example. *Computer Graphics Forum (Proceedings of Eurographics)*. 2007;26(3).
27. Templer J. *The Staircase: Studies of Hazards, Falls, and Safer Design*. MIT Press; 1995. Available from: <http://books.google.com/books?id=n8V546i9zGAC>.
28. Singh S, Kapadia M, Reinman G, Faloutsos P. Footstep navigation for dynamic crowds. *Computer Animation and Virtual Worlds*. 2011;22(2-3):151–158. Available from: <http://dx.doi.org/10.1002/cav.403>.
29. Giese A, Latypov D, Amato NM. Reciprocally-Rotating Velocity Obstacles. In: *ICRA. IEEE*; 2014. p. 3234–3241. Available from: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6895053>.
30. Karamouzas I, Guy SJ. Prioritized Group Navigation with Formation Velocity Obstacles. In: *IEEE International Conference on Robotics and Automation (ICRA)*; 2015. .
31. Choi MG, Kim M, Hyun K, Lee J. Deformable Motion: Squeezing into Cluttered Environments. *Comput Graph Forum*. 2011;30(2):445–453. Available from: <http://dx.doi.org/10.1111/j.1467-8659.2011.01889.x>.
32. van Basten BJH, Stuvell SA, Egges A. A hybrid interpolation scheme for footprint-driven walking synthesis. *Graphics Interface*. 2011;p. 9–16.
33. Brscic D, Kanda T, Ikeda T, Miyashita T. Person tracking in large public spaces using 3-d range sensors. *Human-Machine Systems, IEEE Transactions on*. 2013;43(6):522–534.
34. Fiorini P, Shiller Z. Motion Planning in Dynamic Environments using Velocity Obstacles. *The International Journal of Robotics Research*. 1998 July;17(7):760–762.
35. Best A, Narang S, Manocha D. *Interactive and Conservative Collision Avoidance for Elliptical Agents*. Department of Computer Science, University of North Carolina at Chapel Hill; 2015.
36. Zhang J, Klingsch W, Schadschneider A, Seyfried A. Ordering in bidirectional pedestrian flows and its influence on the fundamental diagram. *J Stat Mech*. 2012;2012(02):P02002.



S1 Figure Pedestrian interactions in crossing flows. (A) Experimental Setup by Imanishi et. al. [6]. A pedestrian, the “traverser”, crosses “pedestrians in flow” with density $\rho \in \{0.25, 1, 2\}$ *people/m²* at an angle $\theta \in \{0, 90, 135, 180\}$ deg. (B) Reproduction of the experiment with elliptical agents for density $1\textit{people/m}^2$ and crossing angle 45 deg

Supporting Information

Acknowledgments

The authors would like to thank the Jülich Supercomputing Institute for their continued help in graciously providing their collected data and supplying a definition for the simulation domain. This research has been supported by grants from Boeing and NSF.