Real-time Reciprocal Collision Avoidance with Elliptical Agents

Andrew Best[1], Sahil Narang[1] and Dinesh Manocha[1]
http://gamma.cs.unc.edu/EORCA/

***Abstract*— We present a novel algorithm for real-time collision-free navigation between elliptical agents. Each robot or agent is represented using a tight-fitting 2D ellipse in the plane. We extend the reciprocal velocity obstacle formulation by using conservative linear approximations of ellipses and derive sufficient conditions for collision-free motion based on low-dimensional linear programming. We use precomputed Minkowski Sum approximations for real-time and conservative collision avoidance in large multi-agent environments. Finally, we present efficient techniques to update the orientation to compute collision-free trajectories. Our algorithm can handle thousands of elliptical agents in real-time on a single core and provides significant speedups over prior algorithms for elliptical agents. We compare the runtime performance and behavior with circular agents on different benchmarks.**

I. INTRODUCTION

One of the main issues in multi-robot planning and multi-agent navigation is computing collision-free paths for each robot or agent from its start position to its goal position. This problem arises not only in robotics and AI, but also in computer games, virtual environments, pedestrian dynamics, and simulations of collective behaviors in biology.

In this paper, we mainly address the problem of real-time multi-agent collision avoidance for large environments with hundreds or thousands of agents. Most practical algorithms are based on decentralized methods that tend to compute a path for each robot or agent independently. Moreover, most of these decentralized algorithms represent each agent as a circle in a 2D plane. This simplified disc representation has a number of advantages. Discs are radially symmetric and there are simple solutions to operations, including computing the closest point on a disc to an arbitrary query point or checking two discs for overlap. Many collision-avoidance algorithms based on velocity obstacles [1], [2], [3] compute Minkowski Sums, which is rather trivial for discs. The symmetric properties are also used to design simple rules for flocking [4] or collision-avoidance schemes based on potential fields or repulsive social forces [5].

Despite the computational benefits, the use of discs for multi-agent navigation results in many challenges. In many cases, a disc overestimates the actual profile of the robots that it represents. The geometries of many robots, including humanoids and vehicles, are not radially symmetric. As a result, disc-based collision-avoidance schemes can be overly conservative. In other applications, tighter-fitting geometric shapes (e.g. ellipses) are considered more accurate. For example, elliptical shapes are used in motion planning algorithms for robots with linear dynamics [6] or to represent uncertainty [7], [8]. Prior work in pedestrian dynamics and biomechanics has shown that 2D ellipses provide a better approximation of human body in terms of shape and movement [9], [10].

It is widely accepted that the use of ellipses for multi-agent navigation can result in complex and time-consuming algorithms. The underlying techniques must explicitly model the orientation of each ellipse. Moreover, computing exact Minkowski Sums of ellipses is regarded as considerably more expensive than doing the same for circles [11]. It is difficult to predict the behavior of force-based methods, as the formulation and computation of repulsive forces on elliptical agents is more complicated.

Main Results: We present a novel real-time algorithm for collision-free navigation of elliptical agents in dynamic environments. We use a decentralized approach based on velocity obstacles [2], [3] and present fast algorithms for conservative collision avoidance. To overcome the complexity of exact Minkowski Sum computation, we use conservative linear approximations of ellipses and reduce the collision avoidance problem to solve a lower dimensional linear programming problem and show that our formulation is reciprocally maximal. To ensure that we can handle large environments consisting of hundreds or thousands of agents in real-time, we use precomputed tables of Minkowski Sums and guarantee that feasible trajectories computed using our algorithm would be collision-free. We also present a technique to update the orientation of ellipses in dense environments to compute collision-free trajectories among static obstacles or to generate human-like trajectories. In practice, the use of conservative linear approximations and precomputed Minkowski Sums tables improves the runtime performance by more than an order of magnitude. Our overall algorithm can perform collision-free navigation of thousands of agents at interactive rates on a single core and is only $4-5$X slower than collision avoidance algorithms for circular agents. We highlight its performance in complex environments and demonstrate the benefits.

The rest of the paper is organized as follows. In Section II, we provide a brief overview of prior work in multi-agent navigation and collision avoidance algorithms. We introduce the notation and present our optimal collision avoidance algorithm between elliptical agents in Section III. In Section IV, we present our acceleration scheme based on precomputed Minkowski Sum tables. We describe two

*This work is supported by ARO contract W911NF-14-1-0437, and a grant from the Boeing company.

[1] Andrew Best, Sahil Narang and Dr. Dinesh Manocha are with the department of Computer Science at the University of North Carolina at Chapel Hill, 201 S. Columbia St, Chapel Hill, NC 27599-3175.

different methods to update the orientation of elliptical agents in Section V. We highlight our algorithm's performance in different environments in Section VI.

II. RELATED WORK

There is a substantial body of work on motion planning and navigation for single or multi-robot systems. Some earlier methods were designed for static environments.

At a broad level, prior algorithms for collision-free navigation can be classified as centralized and decentralized. Centralized planners treat the set of all robots as a single system in a (very) high-dimensional configuration space, and many well-known methods for single-robot motion planning can be used [12], [13], [14]. These planners have the advantage of being complete (in theory), but practical algorithms are limited to systems with a few robots. Decentralized planners tend to compute a path for each robot or agent independently, and use some coordination mechanism or local navigation techniques to avoid collisions between them. Many techniques have also been proposed for collision-avoidance, navigation, and planning among moving obstacles [15], [16], [17], [18], and these can be extended to elliptical agents. Given an agent, all the other agents can be treated as dynamic obstacles along with other objects in the scene. However, these methods do not consider the reactive behavior of other agents as part of multi-agent planning.

Many decentralized planners tend to rely on high-level planning modules to generate paths through the static environment and on local collision-avoidance algorithms to adapt those plans to the environment. Priority-based methods assign order to the robots, and plan the paths sequentially [19]. Velocity-obstacle based methods compute locally collision-free velocities in velocity space [1], [20], [2], [3]. Whereas many local collision-avoidance algorithms take advantage of the disc representation of the robots, some algorithms also attempt to model additional dynamics constraints, such as differential-drive [21], double-integrator [22], car-like [23], or linear [24], etc., or they use reciprocally rotating velocity obstacles [25]. The two dimensional disc collision-avoidance problem has also been extensively studied in the crowd simulation and pedestrian dynamics literature. These studies include force-based methods [5], [26], [27], cellular-decomposition methods [28], rule-based methods [4], etc. Although all these algorithms are fast, they are limited to circular agents.

III. MULTI-AGENT COLLISION AVOIDANCE WITH ELLIPTICAL AGENTS

In this paper, we address the problem of the efficient navigation of multiple elliptical agents. We first introduce the notation used in the rest of the paper. Let $\mathbb{R}^d$ represent the physical workspace of the robots or agents, where $d \geq 2$. For simplicity, we assume each agent can be represented in $\mathbb{R}^2$. We project the geometric representation $\mathcal{O}^d$ of the robot in $\mathbb{R}^d$ space to $\mathbb{R}^2$, represented by $\mathcal{O}^2$, and bound it with an ellipse in $\mathbb{R}^2$.

A. PROBLEM DEFINITION

Let $\mathbb{S}$ represent the simulator state, defined as the union of all entities in the scene, including obstacles in the environment and the overall state space $\mathbb{X} = \cup_i \mathcal{X}_i$, where $\mathcal{X}_i$ denotes the state space of robot i [29]. The elliptical shape in $\mathbb{R}^2$ for each robot i is defined by the position vector $\vec{p}_i \in \mathbb{R}^2$, orientation $o_i \in \mathbb{R}$, the semi-major axis $s_i^{maj} \in \mathbb{R}^+$, and the semi-minor axis $s_i^{min} \in \mathbb{R}^+$ where orientation is defined as the angle between the major axis and the x-axis of the reference frame. Also, let $\vec{v}_i^0$ represent the optimal velocity toward the goal with respect to static obstacles (also called the *preferred velocity*) and let o^0 denote the preferred orientation for the robot. Then the state space of robot i in $\mathbb{R}^{10}$ is given by $[\vec{p}_i, \vec{v}_i, \vec{v}_i^0, o_i, o_i^0, s_i^{maj}, s_i^{min}]$.

We assume that there is a high-level module $\mathcal{K}_i : t \times \mathbb{S} \rightarrow \mathbb{R}^2$ that maps the time t and simulator state $\mathbb{S}$ into an instantaneous preferred velocity that can be expressed as the composition of simpler functions such as

$$\mathcal{K}_i(t) = \mathcal{P}_i(\mathcal{G}_i(t)), \tag{1}$$

where $\mathcal{G}_i : t \times \mathbb{S} \rightarrow \mathbb{R}^2$ maps the time and simulator state into a goal position and $\mathcal{P}_i : \mathbb{S} \times \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that maps the simulator state and the agent's goal position into a instantaneous preferred velocity for agent i, denoted by $\vec{v}_i^o$. The function $\mathcal{K}_i$ computes the collision-free path to the goal with respect to static obstacles in the simulation. Let $\mathcal{LCA}_i : \mathbb{S} \times \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}^2 \times \mathbb{R}^2$ denote a local collision avoidance function that maps the simulator state, the instantaneous preferred velocity, the instantaneous preferred orientation, and time horizon, τ, into a collision-free orientation (o_i) and velocity ($\vec{v}_i$) with respect to other robots in the environment for at least time τ.

Our goal is to formulate a generalized local collision avoidance function $\mathcal{LCA}$ for elliptical agents, which seeks to independently and simultaneously compute a velocity $\vec{v}_i$ and orientation o_i for each elliptical robot i in the simulation. Thus, the instantaneous velocity and orientation of an agent can be given by:

$$(\vec{v}_i(t), o_i(t)) = \mathcal{LCA}_i(\mathcal{H}_i(\mathcal{P}_i(\mathcal{G}_i(t)))), \tag{2}$$

where $\mathcal{H}_i : \mathbb{S} \times \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}^2$ maps the simulator state, preferred velocity, and time to the preferred orientation. The function $\mathcal{LCA}$ must guarantee that all elliptical agents can follow a collision-free trajectory with the new configuration for at least a predefined horizon window of time τ. Furthermore, the agent's new velocity should be as close as possible to its preferred velocity.

B. LOCAL COLLISION AVOIDANCE

In this section, we present our local collision-avoidance algorithm for elliptical agents. Our algorithm is based on velocity obstacles that have been frequently used for collision-avoidance in robotics and crowd simulation [3], [2]. However, as with most prior work in multi-agent navigation, these methods are limited to agents represented as 2D discs.

Velocity Obstacles: For two agents, X and Y, centered at $\vec{p}_X$ and $\vec{p}_Y$, respectively, the velocity obstacle of X induced

by Y, which is denoted by $VO^{\tau}_{X|Y}$, and constitutes the set of velocities for X that would result in a collision with Y at some time before τ. By definition, agents X and Y are guaranteed to be collision-free for at least time τ, if $\vec{v}_X - \vec{v}_Y \notin VO^{\tau}_{X|Y}$ [3]. More formally,

$$VO^{\tau}_{X|Y} = \{\vec{v} \mid \exists t \in [0,\tau] :: \vec{p}_X + t(\vec{v}_X - \vec{v}_Y) \in M\}, \quad (3)$$

where $t \geq 0$ and M denotes the Minkowski Sum of $-X$ and Y.

In general, let V denote the set of all 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, which is a sufficient condition for collision-free navigation for at least time τ.

In the case of disc-shaped agents, the Minkowski Sum for two agents can be implicitly computed with a few floating point operations. Finding the closest point on a disc from a query point is also trivial. For elliptical agents, these operations are non-trivial. Computing the Minkowski Sum requires either computing convolution curves, which is considerably more expensive [30], or using a closed-form implicit equation [11]. Moreover, computing the closest points on two arbitrarily oriented ellipses requires computing the roots of a fourth-order polynomial. These operations are costly, so we present faster algorithms based on conservative linear approximations.

C. APPROXIMATING ELLIPSES

Instead of computing the exact Minkowski Sum of two ellipses, we use a piece-wise linear approximation (PL) that can provide conservative guarantees for collision avoidance. For an ellipse $\mathcal{C}$, a piece-wise linear approximation can be computed by uniformly sampling $\mathcal{C}$ at intervals of $\delta\theta \in (0, 90)$ to yield the set of sample points $\mathcal{S} = \{(s_{maj} \times cos(\delta\theta \times i), s_{min} \times sin(\delta\theta \times i) : 0 \leq i \leq \lfloor \frac{2\pi}{\delta\theta} \rfloor | i \in \mathbb{I}\}$. Let $\mathcal{L} = \{\lambda_{\vec{p}} | \vec{p} \in \mathcal{S}\}$ denote the set of tangents to the curve $\mathcal{C}$, defined at each sample point as:

$$\lambda_{\vec{p}} = \mathcal{C}(\vec{p}) + t\mathcal{C}'(\vec{p}) \ \forall \ \vec{p} \in \mathcal{S}, \quad (4)$$

where $t \geq 0$. We can now define the set of vertices $\mathcal{V}$ of the bounding polygon by solving $\lambda_{\vec{p}_A} = \lambda_{\vec{p}_{A+1}}$ i.e., the point of intersection of two tangents where $\vec{p}_A, \vec{p}_{A+1} \in \mathcal{S}$.The computational cost is thus $O(m)$ for m samples. Next, we use this property to show that the linear approximation computes a conservative shape for collision avoidance.

Theorem 1: *For any ellipse $\mathcal{C}$, a piecewise linear approximation $\mathcal{L}$ of the curve defined by the tangents at the sampled points overestimates the curve.*

Proof: Let $\mathcal{C}$ denote an axis-aligned ellipse defined at the origin as:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1, \quad (5)$$

where $a^2 > b^2$ and $a, b \in \mathbb{R}^+$. The first and second derivative of the curve, $\dot{\mathcal{C}}$ and $\ddot{\mathcal{C}}$ resp., are given by:

$$\frac{dy}{dx} = -\frac{b^2 x}{a^2 y}, \quad (6)$$

$$\frac{d^2 y}{dx^2} = -\frac{b^4}{a^2 y^3}. \quad (7)$$

It follows that:

$$\begin{cases} \ddot{\mathcal{C}} > 0 & \text{if } y < 0, \\ \ddot{\mathcal{C}} < 0 & \text{if } y > 0. \end{cases}$$

By definition, $\mathcal{C}$ is concave downward for $y > 0$, which implies that the tangent lines $\lambda_{\vec{p}} \in \mathcal{L}$ lie above the curve when $\vec{p}$ lies in the first or second quadrant. Similarly, $\mathcal{C}$ is concave upward for $y < 0$, which implies that the tangent lines $\lambda_{\vec{p}} \in \mathcal{L}$ lie below the curve in the interval when p lies in the third or fourth quadrant. Hence, the linearization $\mathcal{L}$ overestimates the curve $\mathcal{C}$. Without loss of generality, the same proof can be used for oriented ellipses.

Theorem 2: *We are given an ellipse $\mathcal{C}$ in the form $y = \mathcal{C}(x)$ and a piecewise linear approximation $\mathcal{L}$ of the curve defined by the tangents at the ordered set of samples $\mathcal{S}$. Let $\lambda_A \in \mathcal{L}$ denote a linearization centered at $\vec{p}_A = (x_A, \mathcal{C}(x_A))$ that approximates the curve in the interval $\mathcal{I} = (\vec{p}_A, \vec{p}_j)$ where $\vec{p}_A = (x_A, \mathcal{C}(x_A))$ and $\vec{p}_j = (x_j, \mathcal{C}(x_j))$. Then the error bounding function $B(t)$ over $\mathcal{I}$ can be given by :*

$$\mathcal{B}(x_s) = \frac{K}{(1 - \frac{x_s^2}{a^2})^{3/2}}, \quad (8)$$

where $K = \frac{b(x_A - x_j)^2}{2a^2}$ and $x_A < x_S < x_j$. Furthermore, the maximum approximation error, $\mathcal{E}_{max}$, can be exactly computed.

Proof: As before, let us consider an axis-aligned ellipse defined at the origin. We also assume that the set of samples $\mathcal{S}$ includes the points where $y = 0$. Let $\mathcal{E}(x_j)$ represent the approximation error for point $(x_j, \mathcal{C}(x_j))$, which is expressed as:

$$\mathcal{E}(x_j) = \mathcal{C}(x_j) - \lambda_A|_{x=x_j} \quad (9)$$

$$\lambda_A|_{x=x_j} \geq \mathcal{C}(x_j) \, \forall \, x_j \in \mathbb{R} \quad (Theorem 1) \quad (10)$$

$$\therefore \mathcal{E}(x_j) \leq 0 \quad (11)$$

It can be proven that there is some $x_s \in \mathcal{I}$ such that:

$$\mathcal{E}(x_j) = \frac{\ddot{\mathcal{C}}(x_s)}{2}(x_j - x_A)^2. \quad (12)$$

We can derive an expression for the error-bounding function using 5 , 7, & 11, as:

$$B(x_s) = |\mathcal{E}(x_j)| = \frac{K}{(1 - \frac{x_s^2}{a^2})^{3/2}} \quad (13)$$

Differentiating $B(x_s)$ gives:

$$\dot{B}(x_s) = \frac{3K}{a^2(1 - \frac{x_s^2}{a^2})^{\frac{5}{2}}} x_s. \quad (14)$$

It follows that the error bound is monotonically increasing when $x_s > 0$ i.e. $x_A, x_j \in \mathbb{R}^+$. Similarly, it is monotonically

decreasing when $x_s < 0$ i.e. $x_A, x_j \in \mathbb{R}^-$. Thus the maximum approximation error, $\mathcal{E}_{max}$, over the open interval $\mathcal{I} = (\vec{p}_A, \vec{p}_j)$ can be expressed as:

$$\mathcal{E}_{max} = \begin{cases} \frac{3K}{a^2(1-\frac{x_s^2}{a^2})^{\frac{5}{2}}} x_s \Big|_{x_s = x_j} & \text{if } x_A, x_j \in \mathbb{R}^+, \\ \frac{3K}{a^2(1-\frac{x_s^2}{a^2})^{\frac{5}{2}}} x_s \Big|_{x_s = x_A} & \text{if } x_A, x_j \in \mathbb{R}^-. \end{cases}$$

D. VELOCITY OBSTACLES FOR ELLIPTICAL AGENTS

In this section, we extend the ORCA algorithm [2] to elliptical agents based on the linear approximation. We refer to the new algorithm as ERVO.

1) Computing Neighboring Agent Constraints: To compute the velocity obstacle for an elliptical agent, we first compute a tangent from the origin of the velocity space to the boundaries of the Minkowski Sum scaled by the inverse of τ. For a Minkowski Sum with m samples, we can find tangents in $O(lg(m))$ using binary search on the vertices. The forward face of the truncated cone lies between the tangent points, and the nearest point can be computed by another binary search. Fig 1(A, B) illustrates the construction of the velocity obstacle of the elliptical agents using the Minkowski Sum and the tangents.

Next, we use the velocity obstacle to compute the set of permitted velocities for an agent. Given the velocity obstacle, $VO^{\tau}_{X|Y}$, we construct the set of permitted velocities for X for reciprocal collision avoidance [2], which is denoted as $\text{ERVO}^{\tau}_{X|Y}$. Consider a vector $\vec{u}$ from the relative velocity $\vec{v}_X - \vec{v}_Y$ to the nearest point on the truncated velocity obstacle. Also, let $\vec{n}$ be the outward normal of the boundary of $VO^{\tau}_{X|Y}$ at point $(\vec{v}_X - \vec{v}_Y) + \vec{u}$. We assume that all agents use the same collision-avoidance strategy. Therefore, agent X is responsible for adapting its velocity by $\frac{1}{2}\vec{u}$ assuming that Y will do the same. In this manner, the set $ERVO^{\tau}_{X|Y}$ of permitted velocities for X is defined by the half-plane pointing in the direction of $\vec{n}$ centered at the point $\vec{v}_X + \frac{1}{2}\vec{u}$ (see Fig 1(C)). Hence, one half-plane constraint is constructed for each neighboring agent.

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. To compute the velocity obstacle $VO^{\tau}_{X|O}$ for agent X induced by a line segment O, we implicitly compute the Minkowski Sum of $-X$ and O scaled by the inverse of τ. An agent X will collide with obstacle O within the time τ if its velocity $\vec{v}_X$ is inside $VO^{\tau}_{X|O}$. Geometrically, the Minkowski Sum is equivalent to sliding the reflected A along the scaled line segment (each scaled by τ). Because discs are unaffected by orientation issues, the computations related to determining the shape of the velocity obstacle, finding tangents, and determining the closest point on the obstacle can be implemented by using a small number of floating point operations. With elliptical agents, the shape of the velocity obstacle, the tangents, and the nearest point operations are governed by the orientation of the agent as well as of the obstacle.

Let $-\theta \in [0, -2\pi]$ represent the angle between O and the positive x-axis. We can simplify the computation by rotating the coordinate system by θ, i.e., we rotate O and agent X by θ and compute the appropriate constraints in the transformed space. Let o_X^{rot} denote the orientation of the ellipse after the rotation transformation. For the remainder of this section, we can assume that O is parallel to the positive x-axis. Therefore, the shape of the velocity obstacle depends only on the orientation o_X^{rot} of the agent. We can also accelerate the computation of tangents and the closest points (Section IV).

Once rotated, we determine whether the agent's velocity projects onto the left tangent, right tangent, left face, right face, or line segment. Next, we compute the point $F_{X|O}$ on the velocity obstacle closest to $\vec{v}_X$. The half plane defined using the tangent to the velocity obstacle at $F_{X|O}$ yields the set $\text{ERVO}^{\tau}_{X|O}$ of permitted velocities for X with respect to O (Figure 1(E)). We rotate the final computed constraint by $-\theta$ to transform back into the original space(Figure 1(F)).

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 neighboring agents and obstacles can be computed efficiently by using a spatial data structure, such as a kD-tree. The set of permitted velocities for agent X is simply the convex region – ERVO^{τ}_X – given by the intersection of the half-planes of the permitted velocities that are induced by all the neighboring agents and obstacle(Figure 1(F)).

$$\text{ERVO}^{\tau}_X = \bigcap_{Y \neq X} \text{ERVO}^{\tau}_{X|Y} \tag{15}$$

The agent is responsible for selecting a new velocity v_X^{new} from ERVO^{τ}_X that minimizes the deviation from its preferred velocity $\vec{v}_X^0$.

$$\vec{v}_X^{new} = \underset{\vec{v} \in \text{ERVO}^{\tau}_X}{\arg\min} \; \|\vec{v} - \vec{v}_X^0\| \tag{16}$$

Eq. 15 and 16 can be solved efficiently with an expected runtime of $O(n)$ using linear programming, where n is the total number of constraints.

4) Collision-free Guarantees:: If the linear programming algorithm can compute a feasible solution for each agent, we can guarantee that the resulting trajectories will be collision-free. This follows from our conservative, bounding linear approximation (Section 3.2.1) for each ellipse. Furthermore, we extended the original ORCA algorithm [2] to elliptical agents by formulating appropriate constraints (Sections 3.2.2 and 3.2.3) that preserve the properties of the velocity obstacles. The set of permitted velocities $ERVO^{\tau}_{X|Y}$ and $ERVO^{\tau}_{Y|X}$ for agents X and Y, respectively, are reciprocally maximal. The overall approach is conservative, but it guarantees collision-free navigation. In densely packed conditions, the ERVO set of feasible velocities may be empty, in which case the 2D linear program will not find a solution. One possibility is to change the orientation of the ellipses (see ERVO-F in Section V) to find a solution. If that does not find a feasible solution, we can select the velocity that minimally penetrates the constraints generated by neighboring robots, by using 3D linear programming [2].

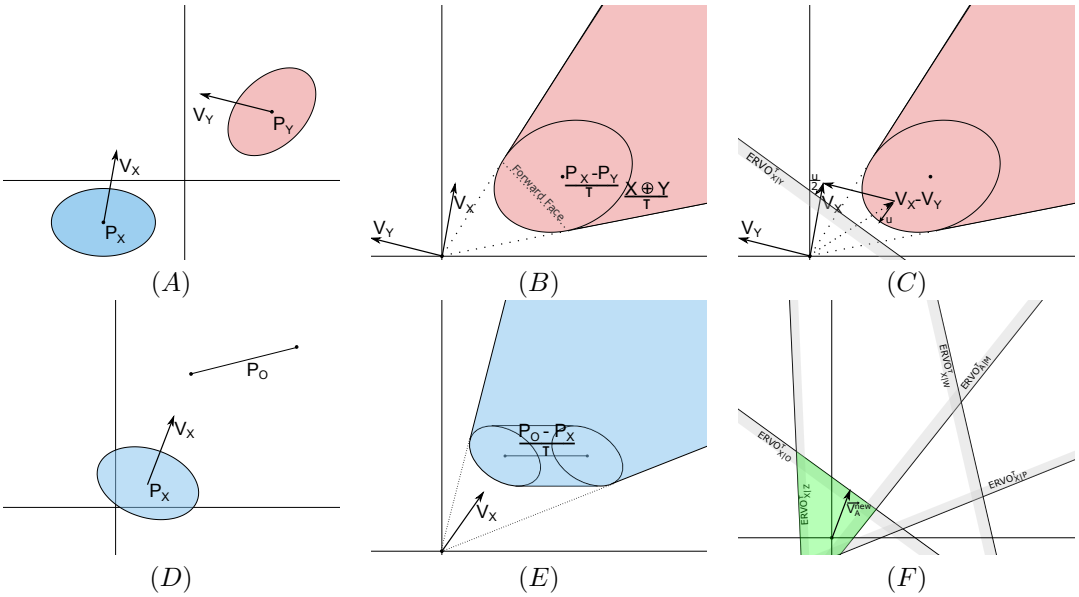


Fig. 1. **COMPUTING COLLISION FREE VELOCITY** (A) Two agents are moving toward one another in space. The approximating polygons are shown. (B) The velocity obstacle for X induced by Y takes the shape of a truncated cone. The apex of the cone is located at the origin in velocity space. The forward arc of the cone is the forward face of the Minkowski Sum of $-X$ and Y scaled by τ and centered at $\vec{p}_Y - \vec{p}_X/\tau$. (C) The set of permitted velocities for agent X w.r.t. Y represented by the half-plane constraint $ERVO^{\tau}_{X|Y}$. (D) An agent moves toward a line segment obstacle. (E) To compute $VO_{X|O}$, we rotate the frame of reference such that O is parallel to the positive x-axis. In this case, the forward arc of the cone is the forward face of the Minkowski Sum of $-X$ and O scaled by τ and centered at $\vec{p}_O - \vec{p}_X/\tau$. We also show the ERVO constraint as a half-plane perpendicular to the vector connecting v_X and $VO_{X|O}$ and passing through the nearest point on $VO_{X|O}$. After the constraint is computed, it is rotated back to the axis-aligned reference frame. (F) After all the constraints have been computed, we can determine a feasible velocity inside the union of all the ERVO sets. A new velocity is chosen from the region of feasible velocities.

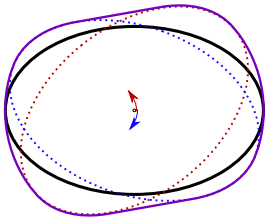


Fig. 2. **Swept Ellipse:** An ellipse (shown in black) swept along an interval with padding in the positive direction (in red) and negative direction (in blue) equal to the agent's maximum angular velocity. The convex-hull of this swept surface (in purple) is used in place of the agent to construct Minkowski Sums and produce collision-free rotations for the ERVO-F algorithm. A table of these swept surfaces is maintained for acceleration of the ERVO-F algorithm.

IV. Accelerating ERVO With Precomputation

The ERVO algorithm described above can compute collision-free trajectories for elliptical agents. However, in practice it is computationally two orders of magnitude times more expensive compared to the original ORCA algorithm for circular agents (see Table I). The main bottleneck is the computation of the Minkowski Sum of the polygons and determining the forward arc on its boundary while constructing the half-plane constraints for each neighboring agent and obstacle.

In order to accelerate the computation, we use precomputed Minkowski Sums for different orientations of ellipses and still guarantee collision-free navigation. In particular, we use a precomputed table of Minkowski Sums such that the runtime computation is $O(1)$, corresponding to a table lookup. We use a discrete angular resolution of θ_E for this precomputation. Let $\mathcal{P} = \{\theta_E \times i : 0 \leq i \leq \lfloor \frac{2\pi}{\theta_E} \rfloor | \theta_E \in (0, 2\pi)\}$ denotes an ordered set of angles. Also, let $\mathcal{K} = \{ROT(X(\theta_i), \theta_{i+1}))|\theta_i, \theta_{i+1} \in \mathcal{P}\}$ denote the set of surfaces generated by rotating an ellipse X from orientation θ_i to θ_{i+1} for each ordered pair (θ, θ_{i+1}) in $\mathcal{P}$. Likewise, let $\mathcal{K}'$ denote the set of swept volumes (Figure 2) generated by rotating an ellipse at orientation θ_i by $\theta_{alpha} = \pm\delta t \times \alpha_{max}$, where δt and α_{max} denote the simulation timestep and maximum angular acceleration of the robot, respectively. Each surface in $\mathcal{K}$ and $\mathcal{K}'$ is parametrized by the angle θ_i.

We precompute and store the pairwise Minkowski Sums for surfaces in $\mathcal{K}$, as well as $\mathcal{K}'$. Each such Minkowski Sum is parameterized by the corresponding pair of angles, corresponding to the two ellipses. Additionally, we also store the extreme points of the polygon which facilitates the construction of tangents and reduces the complexity of finding the closest point on the Velocity Obstacle. Consider an elliptical agent X with orientation θ_X. The precomputed table can be used to find the surface $\mathcal{K}_X$ s.t. $\theta_i \leq \theta_X \leq \theta_{i+1}$ where $\theta_i, \theta_{i+1} \in \mathcal{P}$. By definition, the ellipse representing the agent X is contained within $\mathcal{K}_X$ and is thus, conservative. When computing the constraints for two agents X and Y, we lookup the surfaces $\mathcal{K}_X$ and $\mathcal{K}_Y$ resp., and the corresponding Minkowski Sum. The collision-free guarantees (Section III-B) still hold but the solution may not be optimal since the ERVO sets of collision-free velocities are not maximal.

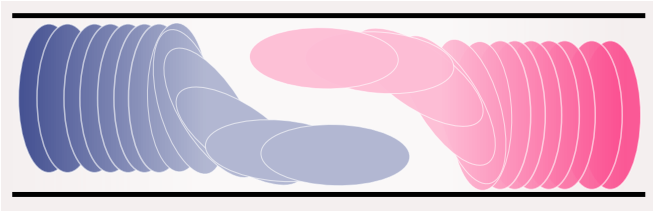


Fig. 3. **Narrow Passage:** Two ellipses approach one another in a narrow hallway. In order to pass safely, each ellipse must rotate to reduce their profile. These behaviors are not possible with disc-based agents and are demonstrated with our ERVO-F algorithm.

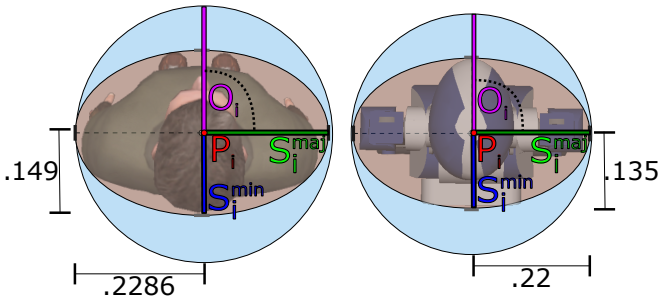


Fig. 4. **Bounding Disc vs Ellipse.** Disc (Blue) and ellipse (Brown) for (Left) a human of average body width and depth; (Right) HRP-4 robot.

Overall, our precomputed structures with an interval size of 5 degrees provide a 40x improvement in performance. The precomputation time is on the order of minutes and spatial complexity is $O(m * o^2)$ where m is the number of samples on the ellipse and o is the number of orientation intervals ($120MB$ for $o = 72, m = 100$).

V. ORIENTATION UPDATE

The use of ellipses increases the configuration space of each to three dimensions - $[x, y, \theta]$. In order to maintain interactive simulation, we decompose the problem in two parts: update the orientation of each ellipse and compute the optimal collision-free velocity for that orientation. By using swept surfaces in place of rotating ellipses for the construction of neighboring agent constraints, we ensure collision-free rotations as agents navigate. Figure 2 illustrates the swept surfaces. We present two simple approaches to orientation computation, given as function definitions $\mathcal{H}$, for orientation update:

- **ERVO-C:** In the simplest case, the agents maintain their orientation between successive simulation time-steps:

$$o_i^0 = o_i. \tag{17}$$

- **ERVO-F:** In some cases, the agent must change its orientation in order to find a collision-free velocity. For example, in Figure 3, the agent must change its orientation to navigate through a narrow hallway. In this method, we determine whether agent i can travel along its current velocity $\vec{v}_i$ with its current orientation o_i by estimating the scalar space, c_i, available to the agent w.r.t its current orientation at a point slightly ahead in the direction of travel. Let $\mathcal{MC}(o)$ denote the minimum clearance required for an agent with orientation o. Then, o_i^0 is given by:

$$o_i^0 = \begin{cases} o_i & \text{if } c_i \geq \mathcal{MC}(o_i), \\ o_i^E & \text{otherwise.} \end{cases}$$

  where o_i^E denotes closest orientation to o_i such that $c_i \geq \mathcal{MC}(o_i^E)$. Clearance can be computed as the distance to the nearest point on the nearest neighboring agent or obstacle from the extreme point on either side of the agent with respect to its velocity.

Simulation Update At each time step, we evaluate $\mathcal{H}$ and $\mathcal{K}$ (Section IV) to determine the preferred orientation o^0 and preferred velocity $\vec{v}^0$ for robot. Next, we compute the ERVO constraints (Section III-D) using "swept" surfaces from the set $\mathcal{K}'$ for agents for which $o \neq o^0$ and the less conservative surfaces from the set $\mathcal{K}$ for the remaining ones. The agents for which the 2D linear program computes a feasible solution can update their orientation. The remaining robots maintain their orientation and we use 3D linear programming to compute the "safest possible" velocity.

VI. RESULTS

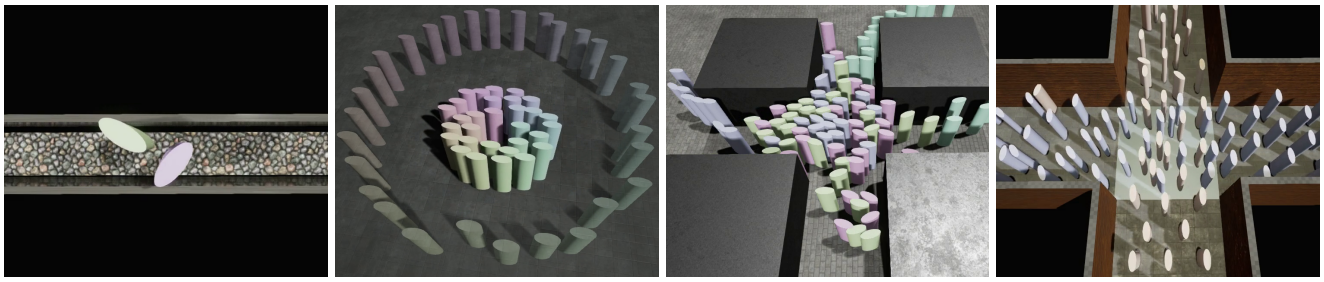
In this section, we highlight the performance of our algorithm on several planar navigation benchmarks (Figure 5). We model a human body with average width and depth ($s^{maj} = .2286$, $s_{min} = .149$, $r = .2286$) for our experiments, and an HRP-4 humanoid robot ($s^{maj} = .22$, $s_{min} = .135$, $r = .22$) for the crossflow scenario.

We set the sampling size at $m = 100$ points for our piece-wise linear approximation and orientation intervals of 5 degrees for precomputed surfaces. Using Theorem 2 (Section III-C), the maximum approximation error for a point on the bounding ellipse for a human was found to be 0.005. The aggregate error, defined as the difference between the area of the exact ellipse and approximated ellipse, was $0.0002m^2$.

We implemented our algorithm in C++ on a windows 7 desktop PC. Timing results (Table I) were generated on an Intel i7-4790 pc with $16GB$ of ram. Although both ERVO and ORCA can be parallelized, results were generated on a single core.

A. ERVO Performance: Benefit of Pre-computation

We have evaluated the performance of the ERVO algorithm described in Section III and the acceleration structure that pre-computes the Minkowski Sums in Section IV. In particular, we compared the performance of ERVO, with and without precomputation, to compute collision-free trajectories of a large number of elliptical agents and also compared them with ORCA, which uses circular agents, on the antipodal benchmark. In this scenario, the agents are initialized on a circle and their goal position is set to the antipodal position. We plot the average frame update time as function of the number of agents in Figure 6.



(A) (B) (C) (D)

Fig. 5. **Experiment Scenarios.** (A) Two agents approach and rotate in the narrow hallway (B) 30 agents cross the antipodal circle (C) 100 Agents cross through the center of the 4-square scene (D) Two groups of 44 HRP-4 form-factor agents cross one-another.

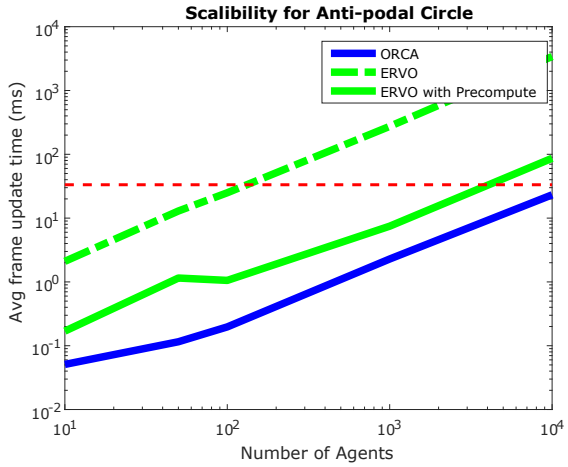


Fig. 6. **Relative Performance of ORCA and ERVO (with and without precomputation).** The average frame update time for ERVO with precomputation is 4-5x slower than ORCA but still interactive for 1000's of agents, shown by the red line denoting 30 FPS. It is also two orders of magnitude faster than ERVO without precomputation.

B. Narrow Benchmarks

In order to demonstrate the benefits of using elliptical agents, we used some benchmarks with narrow passages, where the circular agents would not be able to find a collision-free path.

Narrow Hallway: In this benchmark, an agent must navigate through a narrow hallway of width .4 meters in order to reach its goal position, which requires the agent to rotate and change its orientation. In this case, ERVO-F is able to easily compute collision-free trajectories. ERVO-C can only work if the default orientation is set properly.

Hallway Headon: In this benchmark, two agents approach each other in a narrow hallway of width 0.7 meters i.e. agents must rotate to pass through (Figure 5(A)). In this scenario, the ERVO-F agents change their orientation to compute a collision-free trajectory. However, ORCA agents cannot find a collision-free solution.

C. Other Benchmarks

We also demonstrate the performance of our algorithm on several benchmark scenarios. For each scenario, we compare the total simulation time, average frame update time, and the number of collisions for ERVO-C, ERVO-F and ORCA (Table I).

Scene	Num. Agents	Model	Sim Time (s)	Avg. Update Time (ms)	Avg. Collisions (m)
Circle	100	ORCA	38.19	0.1972	8.9e-05
		ERVO-C	33.54	1.056	8.1e-04
		ERVO-F	27.5	5.38	7.4e-04
Four-Square	100	ORCA	54.52	0.2918	3.05e-07
		ERVO-C	55.56	1.1012	7.52e-05
		ERVO-F	53.13	3.95608	9.82e-05
HRP-4 Crossflow	88	ORCA	29.7	0.233	0
		ERVO-C	29.58	0.899	1.55e-06
		ERVO-F	29.64	3.954	0

TABLE I

COMPARISON RESULTS. ERVO REDUCES TOTAL SOLUTION TIME WITH A SMALL INCREASE IN COMPUTATION TIME (APROX. 5X).

Antipodal Circle: This benchmark (Figure 5 B, Figure 6) is commonly used by prior multi-agent navigation algorithms [2]. The scene demonstrates the ability to perform collision avoidance at very high densities as the agents pass through the center of the circle (Figure 5(B)). ERVO agents are able to navigate the circle 40% faster than ORCA agents.

Four-Square: In this scene, four groups of agents cross through a constrained space between four large square obstacles. This scenario demonstrates ERVO's capability to utilize space more effectively, and thus, decreases the overall simulation time that corresponds to all the agents reaching their goal positions (Figure 5(C)).

HRP-4 Crossflow: In this benchmark, the agents are configured to the dimensions of the HRP-4 humanoid robot platform ($s^{maj} = 0.22m$, $s^{min} = 0.135m$). We demonstrate two groups of agents walking in opposing directions through a hallway (Figure 5(D)). ERVO-F agents are able to change their orientation and navigate through the high density region at the crossing region.

It is evident from our results (Table I) that both ERVO-C and ERVO-F are computationally comparable to ORCA. However, ERVO-F significantly reduces the solution time as agents utilize the space more effectively by changing their orientation. The average collision, measured as the average of the interval penetration depth at each time step, is negligible in each case.

VII. CONCLUSIONS AND FUTURE WORK

We have presented a novel algorithm for reciprocal collision-avoidance between multiple elliptical agents. We use a piecewise linear approximation of ellipses and reduce the velocity computation problem to linear programming. Furthermore, we use a precomputed Minkowski Sum table to reduce the runtime overhead and present techniques to update the orientation. We demonstrate our ability to simulate hundreds of elliptical agents at interactive rates and provide orientation computation and guaranteed collision-avoidance. Furthermore, we show the benefits of using elliptical agents over disc agents in narrow benchmarks and high density scenarios.

Our work has several limitations. Firstly, the orientation update is decoupled from the velocity computation, and we would like to simultaneous optimize both the velocity and the orientation. Second, the ERVO algorithm without precomputation is expensive compared to disc-based methods and the choice of an appropriate orientation interval is not easy. Third, our algorithm assumes perfect sensing and does not account for hardware related errors. In terms of future work, we would like to explore improved strategies for updating the orientation and also take into account dynamics-related constraints in trajectory computation. We would like to validate the accuracy of our method by comparing the trajectory results with those of human crowds. We would also like to test the applicability of our algorithm on physical robots in real environments.

REFERENCES

[1] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008, pp. 1928–1935.

[2] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Inter. Symp. on Robotics Research*, 2011, pp. 3–19.

[3] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[4] C. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proc. of SIGGRAPH*, 1987.

[5] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, no. 5, pp. 4282–4286, May 1995.

[6] D. J. Webb and J. van den, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5054–5061.

[7] N. E. Du Toit and J. W. Burdick, "Robotic motion planning in dynamic, cluttered, uncertain environments," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 966–973.

[8] J. Van Den Berg, P. Abbeel, and K. Goldberg, "Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.

[9] M. Gérin-Lajoie, C. L. Richards, and B. J. McFadyen, "The negotiation of stationary and moving obstructions during walking: Anticipatory locomotor adaptations and preservation of personal space," *Motor Control*, vol. 9, pp. 242–269, 2005.

[10] M. Chraibi, A. Seyfried, and A. Schadschneider, "Generalized centrifugal-force model for pedestrian dynamics," *Phys. Rev. E*, vol. 82, no. 4, p. 046111, 2010.

[11] Y. Yan and G. S. Chirikjian, "Closed-form characterization of the minkowski sum and difference of two ellipsoids," *Geometriae Dedicata*, pp. 1–26, 2014.

[12] S. M. LaValle, S. Hutchinson, *et al.*, "Optimal motion planning for multiple robots having independent goals," *Robotics and Automation, IEEE Transactions on*, vol. 14, no. 6, pp. 912–925, 1998.

[13] G. Sanchez and J.-C. Latombe, "Using a prm planner to compare centralized and decoupled planning for multi-robot systems," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 2112–2119.

[14] P. Švestka and M. H. Overmars, "Coordinated path planning for multiple robots," *Robotics and autonomous systems*, vol. 23, no. 3, pp. 125–152, 1998.

[15] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[16] D. Hsu, R. Kindel, J. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," vol. 21, no. 3, pp. 233–255, 2002.

[17] L. Jaillet and T. Simeon, "A PRM-based motion planner for dynamically changing environments," 2004, pp. 1606–1611.

[18] S. Petty and T. Fraichard, "Safe motion planning in dynamic environments," 2005, pp. 3726–3731.

[19] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 1-4, pp. 477–521, 1987.

[20] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. C. Lin, D. Manocha, and P. Dubey, "Clearpath: Highly parallel collision avoidance for multi-agent simulation," in *ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION*. ACM, 2009, pp. 177–187.

[21] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, *Optimal reciprocal collision avoidance for multiple non-holonomic robots*. Springer, 2013.

[22] E. Lalish and K. A. Morgansen, "Distributed reactive collision avoidance," *Autonomous Robots*, vol. 32, no. 3, pp. 207–226, 2012.

[23] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart, "Reciprocal collision avoidance for multiple car-like robots," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 360–366.

[24] D. Bareiss and J. van den Berg, "Reciprocal collision avoidance for robots with linear dynamics using lqr-obstacles," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3847–3853.

[25] A. Giese, D. Latypov, and N. M. Amato, "Reciprocally-rotating velocity obstacles," in *ICRA*. IEEE, 2014, pp. 3234–3241. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6895053

[26] N. Pelechano, J. Allbeck, and N. Badler, "Controlling individual agents in high-density crowd simulation," in *Symposium on Computer Animation*, 2007, pp. 99–108.

[27] I. Karamouzas, P. Heil, P. van Beek, and M. H. Overmars, "A predictive collision avoidance model for pedestrian simulation," in *Motion in Games*, ser. Lecture Notes in Computer Science, vol. 5884. Springer, 2009, pp. 41–52.

[28] A. Schadschneider, "Cellular automaton approach to pedestrian dynamics - theory," *Pedestrian and Evacuation Dynamics*, pp. 75–86, 2002.

[29] S. Curtis, A. Best, and D. Manocha, "Menge: A modular framework for simulating crowd movement," Department of Computer Science, University of North Carolina at Chapel Hill, http://gamma.cs.unc.edu/menge/, Tech. Rep., 2014.

[30] I.-K. Lee, M.-S. Kim, and G. Elber, "Polynomial/rational approximation of minkowski sum boundary curves," *Graphical Models and Image Processing*, vol. 60, no. 2, pp. 136–165, 1998.