CMSC 425: Lecture 20
Motion Planning: Crowd Motion

Reading: The material on velocity obstacles is taken from J. van den Berg, S. Guy, M. C. Lin, D. Manocha, "Reciprocal n-body Collision Avoidance," *Proc. Int. Symposium of Robotics Research* (ISRR), 2009.

Overview: So far, we have discussed path planning and motion planning in games from various perspectives, ranging from point-to-point motion of a single agent to the coordinated movement of flocks of birds. Today, we will discuss an important problem, which arises in many games. This is simulating the behavior of humans in crowded situations.

Crowd Motion: Today, we will discuss motion simulation involving a number of intelligent autonomous agents, as arises in crowds of pedestrians. Unlike flocking systems, in which it is assumed that the agents behave homogeneously, in a crowd it is assumed that each agent has its own agenda to pursue (see Fig. 1). For example, we might imagine a group of students walking through a crowded campus on their way to their next classes. Since such agents are acting within a social system, however, it is assumed that they will tend to behave in a manner that is consistent with social conventions. (I don't want you to bump into me, and so I will act in a manner to avoid bumping into you as well.)

Fig. 1: Crowd simulation.

Crowd simulation is actually a very broad area of study, ranging from work in game programming and computer graphics, artificial intelligence, social psychology, and urban architecture (e.g., planning evacuation routes). In order to operate in the context of a computer game, such a system needs to be decentralized, where each member of the crowd determines its own action based on the perceived actions of other nearby agents. The problem with applying a simple boid-like flocking behavior is that, whereas flocking rules such as alignment naturally produce systems that avoid collisions between agents, the diverse agendas of agents in crowds naturally brings them directly into collisions with other agents (as in pedestrians traversing a crosswalk from both sides). In order to produce more realistic motion, the agents should anticipate where nearby agents are moving, and then plan their future motion accordingly.

We discuss two models of crowd behavior, social-force dynamics and (reciprocal) velocity obstacles.

Social-Force Dynamics: In our presentation of flocking behavior with Boids, we presented a model in which the motion of each artificial animal is determined by a collection of simple local forces based on the other agents in the system. In much the same manner that a physicist would simulate the motion of a collection of particles in computational fluid dynamics, we can simulate the fluid-like motion of animals in the flock. This can be applied to human crowd behavior as well. At an individual level, human behavior is quite chaotic, and it is not easy to predict future motion based on past motion. However, for many common situations the aggregate behavior of a large group of people can be quite predictable. Examples include people walking along corridors or sidewalks, moving into or evacuating from a building, milling around in a large area (such as people at a party or on the floor of a convention). Here is an interesting observation that behavioral research confirms:

- A flock of geese flying over long distances form an *inverted* "V"-shaped formation (with the apex at the *front* of the flock). This is natural, because birds seek to reduce their wind resistance, which this formation does.
- A small group (3–5) people walking in a wide-open area tend to form a *natural* "V"-shape (with the apex at the *rear* of the group). This is natural, because humans want to be able to see each other and hear each other speak, which this formation permits.

In neither case is there a centralized control that enforces either of these behaviors. They just emerge naturally from the local desires of the participants. Social-force dynamics attempts to model the motion of large crowds of humans in terms of simple local forces that incrementally affect their individual motions.

Recalling our earlier lecture on *agents* in AI, each person in a crowd experiences sensory stimuli, which cause a behavioral reaction that depends on the agent's personal aims and is chosen from a set of behavioral alternatives. The choice depends on a utility maximization, which varies by individual. While each individual may have a (highly unpredictable) utility function, the utility functions of a large group can be modeled by a probability distribution. (E.g., 10% of people are just focused on getting as quickly as possible to their destination, 20% are in no hurry or are strolling together with a friend, and 70% are staring at their mobile devices and they are not paying attention to anything else.)

The physics-based model associates with each agent i and each time instant t a *current position*, denoted $p_i(t)$, a *current velocity* vector, denoted $\vec{v}_i(t)$. Based on the agent's desired path (which resulted from some path planning procedure) each agent has a *target velocity* vector, denoted $\vec{v}_i^{\,0}(t)$. The current velocity may be affected by immediate forces (the desire to avoid a collision, for example), but the target remains relatively stable and points towards the agent's ultimate goal.

As described below, various forces will be evaluated at each time step. Each force will be represented as a vector, and the sum of these forces will result in an *aggregate force* vector, denoted $\vec{F}_i(t)$. (For example, the aggregate force will tend to push the agent away from other agents in the crowd and obstacles in the environment, and will also nudge it back towards the target velocity.)

Given the current state $(p_i(t), \vec{v}_i(t), \vec{v}_i^{\,0}(t))$, and the aggregate force vector $\vec{F}_i(t)$, physics integration can be applied just as in the flocking lecture to simulate the motion of the agents. Here are some issues that affect the forces.

Separation: The agent would like to avoid collisions with other agents in the crowd. We can model this as a *repulsive force* vector $f_{ji}(t)$ that is directed from a nearby agent j toward agent i. This force would presumably be the greatest for other agents that are within a proximity sphere about agent i. Ideally, this force should take into consideration the velocity of the other agent j. (In the discussion below on reciprocal velocity obstacles, it is discussed how to compute such a force.)

Obstacle Avoidance: Another form of separation is a repulsive force to keep the agent from colliding with obstacles in the scene. (This includes both static objects, such as structures and utility poles, and dynamic objects, such as motor vehicles.)

Attraction: An agent that is walking with a group (e.g., a single companion, within a small group of friends, or as part of a tour group) will want to maintain proximity with other agents within the group. This can be modeled as a force $f_{ig}(t)$ that attracts agent i to the perceived center of group g to which it belongs (or perhaps to the leader of the group in the case of a tour group).

Traffic signals and social conventions: In urban settings, signals like walk-lights and cross-walks naturally induce forces that encourages or discourages movement across roadways. In certain regions, there is a tendency to walk on the right side or left side of a corridor or sidewalk, presumably drawn from the conventions for driving in the region.

Individual variations: Not all individuals respond in the same manner to the same stimuli. When facing a back-up, a passive agent will wait patiently, while a more aggressive agent will try to find a way around the back-up or muscle through the crowd.

Velocity Obstacles: While the social-force dynamics can produce reasonable crowd behavior in some environments, it does not always produce the most accurate motion. This is due in part because humans have the ability to anticipate the movement of nearby agents, and alter their motion accordingly. Based on relative speeds and directions, each agent can determine a set of *forbidden velocities* (represented as a region of some velocity space) that will lead to an imminent collision. The agent then selects a velocity that is close to its desired velocity, but lying outside of the region of forbidden velocities. A robust method for doing this is based on the notion of *velocity obstacles*.

Suppose that an agent a is moving in a crowded environment where many other agents are also moving. We assume that a can perceive the motions of nearby agents and generate rough estimates on their velocities. Agent a is also interested in traveling to some destination, and (based on our path finding algorithm) we assume that a has a *preferred velocity*, v_a^*, which it would assume if there were no other agents to consider.

For the sake of simplicity, we will model agents as circular disks translating in the plane. Consider two such agents, a and b, of radii r_a and r_b and currently positioned at points p_a and p_b, respectively (see Fig. 2(a)). If we assume that agent a moves at velocity v_a, then at time t it will have moved a distance of $t \cdot v_a$ from its initial position, implying that it will be located at $p_a + t \cdot v_a$. We will refer to this as $p_a(t)$.

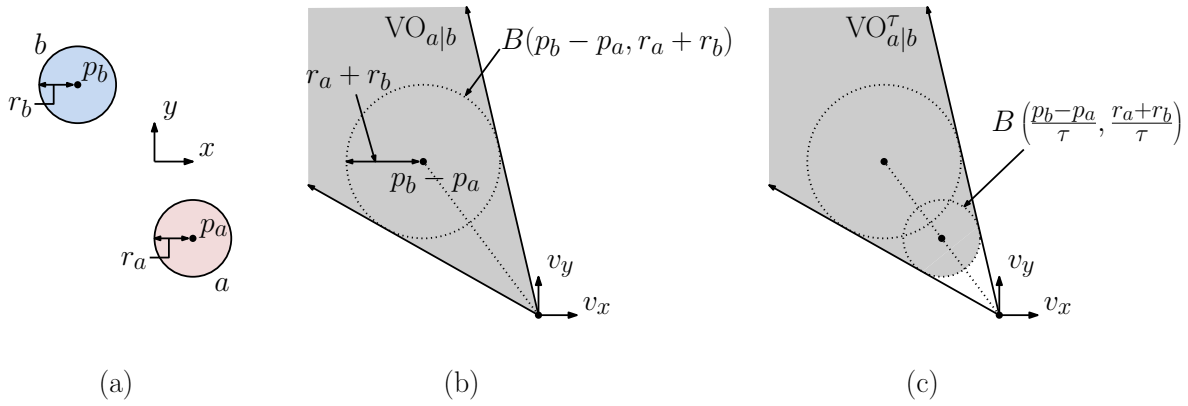

Fig. 2: Velocity obstacle for a induced by b (assuming b is stationary).

For now, let's assume that object b is not moving, that is, $p_b(t) = p_b$ for all t. Let's consider the question of how to select a velocity for a that avoids colliding with b any time in the future. Two disks intersect if the distance between their centers ever falls below the sum of their radii. More formally, let $\|u\|$ denote the Euclidean length of a vector u. Then $\|p_a(t) - p_b(t)\|$ is the distance between p_a and p_b at time t (that is, the length of the vector from b to a). Thus, the two agents collide if $\|p_a(t) - p_b(t)\| < r_a + r_b$. By substituting the definitions of $p_a(t)$ and $p_b(t)$, we find that a velocity v_a will result in a collision some time in the future if there exists a $t > 0$, such that

$$\|(p_a + t \cdot v_a) - p_b\| \;<\; r_a + r_b. \tag{1}$$

We would like to express the velocities v_a that satisfy the above criterion as lying within a certain "forbidden region" of space. To do this, define $B(p, r)$ to be the open Euclidean ball of radius r centered at point p. That is

$$B(p, r) \;=\; \{q : \|q - p\| < r\}.$$

We can rewrite Eq. (1) as

$$\|t \cdot v_a - (p_b - p_a)\| \;<\; r_a + r_b,$$

which is equivalent to saying that $t \cdot v_a$'s distance from the point $p_b - p_a$ is less than $r_a + r_b$, or equivalently, the (parametrized) vector $t \cdot v_a$ lies within a ball of radius $r_a + r_b$ centered at the point $p_b - p_a$. Thus, we can rewrite Eq. (1) as

$$t \cdot v_a \in B(p_b - p_a, r_a + r_b),$$

As t varies from 0 to $+\infty$, the vector $t \cdot v_a$ travels along a ray that starts at the origin and travels in the direction of v_a. Therefore, the set of forbidden velocities are those that lie within a cone that is centered at the origin and encloses the ball $B(p_b - p_a, r_a + r_b)$.

We define the *velocity obstacle* of a induced by b, denoted $\mathrm{VO}_{a|b}$ to be the set of velocities of a that will result in a collision with the stationary object b.

$$\mathrm{VO}_{a|b} \;=\; \{v : \exists t \geq 0,\; t \cdot v \in B(p_b - p_a, r_a + r_b)\}.$$

(See Fig. 2(b).)

One problem with the velocity object is that it considers time going all the way to infinity. In a dynamic setting, we cannot predict the motion of objects very far into the future. So, it makes sense to truncate the velocity obstacle so that it only involves a limited time window. Given a time value $\tau > 0$, what the set of velocities that will result in agent a colliding with agent b at any time t, where $0 < t \leq \tau$ is the *truncated velocity obstacle*:

$$\mathrm{VO}^{\tau}_{a|b} \;=\; \{v : \exists t \in [0, \tau],\; t \cdot v \in B(p_b - p_a, r_a + r_b)\}.$$

This is a subset of the (unbounded) velocity obstacle that eliminates very small velocities (since collisions farther in the future result when a is moving more slowly). The truncated velocity obstacle is a truncated cone, where the truncation occurs at the boundary of the $(1/\tau)$-scaled ball $B((p_b - p_a)/\tau, (r_a + r_b)/\tau)$ (see Fig. 2(c)). Observe that there is an obvious symmetry here. Moving a with velocity v will result in a collision with (the stationary) b if and only if moving b with velocity $-v$ will result in an intersection with (the stationary) a. Therefore, we have

$$\mathrm{VO}^{\tau}_{a|b} \;=\; -\mathrm{VO}^{\tau}_{b|a}.$$

Collision-Avoiding Velocities: Next, let us consider how the velocity obstacle changes if b is moving. If we assume that b is moving with velocity v_b, then a velocity v_a will generate a collision precisely if the differences in their velocities $v_a - v_b$ generates a collision under the assumption that b is stationary, that is, $v_a - v_b \in \mathrm{VO}^{\tau}_{a|b}$. Equivalently, v_a will generate a collision if b if v_a lies in the offset velocity obstacle $\mathrm{VO}^{\tau}_{a|b} + v_b$ (see Fig. 3(a)).

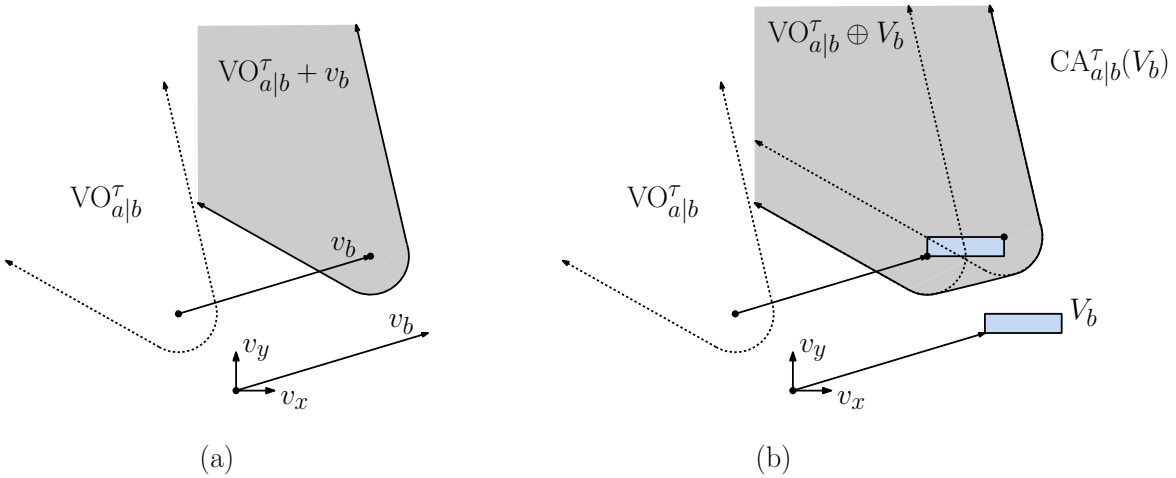

Fig. 3: Velocity obstacles where (a) object b is moving at velocity v_b and (b) object b is moving at any velocity in the set V_b.

We can further generalize this. We usually do not know another object's exact velocity, but we can often put bounds on it. Suppose that rather than knowing b's exact velocity, we know that b is moving at some velocity v_b that is selected from a region V_b of possible velocities. (For example, V_b might be square or circular region in space, based on the uncertainty in its motion estimate.)

Let us consider the velocities of a that *might* result in a collision, assuming that v_b is chosen from V_b. To define this set, we first define the *Minkowski sum* of two sets of vectors X and Y to be the set consisting of the pairwise sums of vectors from X and Y, that is,

$$X \oplus Y \;=\; \{x + y : x \in X \text{ and } y \in Y\}.$$

Then, clearly a might collide with b if a's velocity is chosen from $\mathrm{VO}^{\tau}_{a|b} \oplus V_b$. Therefore, if we want to avoid collisions with b altogether, then a should select a velocity from outside this region. More formally, we define the set of *collision-avoiding velocities* for a given than b selects a velocity from V_b is

$$\mathrm{CA}^{\tau}_{a|b}(V_b) \;=\; \{v : v \notin \mathrm{VO}^{\tau}_{a|b} \oplus V_b\}$$

(see Fig. 3(b).)

Just to recap, if a selects its velocity vector from anywhere outside $\mathrm{VO}^{\tau}_{a|b} \oplus V_b$ (that is, anywhere inside $\mathrm{CA}^{\tau}_{a|b}(V_b)$), then no matter what velocity b selects from V_b, a is guaranteed not to collide with b within the time interval $[0, \tau]$.

This now provides us with a strategy for selecting the velocities of the agents in our system:

- Compute velocity bounds V_b for all nearby agents
- Compute the intersection of all collision-avoiding velocities for these objects, that is

$$\mathrm{CA}^{\tau}_{a} = \bigcap_{b} \mathrm{CA}^{\tau}_{a|b}(V_b)$$

  Any velocity chosen from this set is guaranteed to avoid collisions from now until time τ.
- Select the vector from CA^{τ}_{a} that is closest to a's ideal velocity v^*_a.

In practice, we need to take some care in the implementation of this last step. First, there will be upper limits on fast an object can move or change directions. So, we may not be free to select any velocity we like. Subject to whatever practical limitations we have on what the future velocity can be, we select the closest one that lies within CA^{τ}_{a}. If there is no such vector, then we must consider the possibility that we cannot avoid a collision. If so, we can select a vector that overlaps the smallest number of velocity obstacles.

Issues: While this might seem to be the end of the story with respect to velocity obstacles, there are still issues that arise. One of these issues was hinted at in the last paragraph. In cases where there are lots of agents and the velocity estimates are poor, the collision-avoiding area may be empty. There are a number of strategies that one might consider for selecting a good alternative.

There is a more significant problem with this approach, however, which arises even if we consider only two agents. The problem is that agents that are moving towards each other can engage in a very unnatural looking oscillating motion. The situation is illustrated in Fig. 4. Two agents are moving to each other. They see that they are on a collision course, and so the divert from their initial velocities. Let's imagine the best-case scenario, where they

have successfully resolved their impending collision (whew!) as a result of this diversion (see Fig. 4(b)). Now, each agent sees that the other agent has diverted from its trajectory and reasons, "Great! The other guy has veered off, and I have won the game of chicken. So, now I can resume on my original velocity" (see Fig. 4(c)). So, both agents return to their original velocity, and they are now right back on a collision course (see Fig. 4(d)) and so again they divert (see Fig. 4(e)). Clearly, this vicious cycle of zig-zagging motion will repeat until they manage to make it around one another.

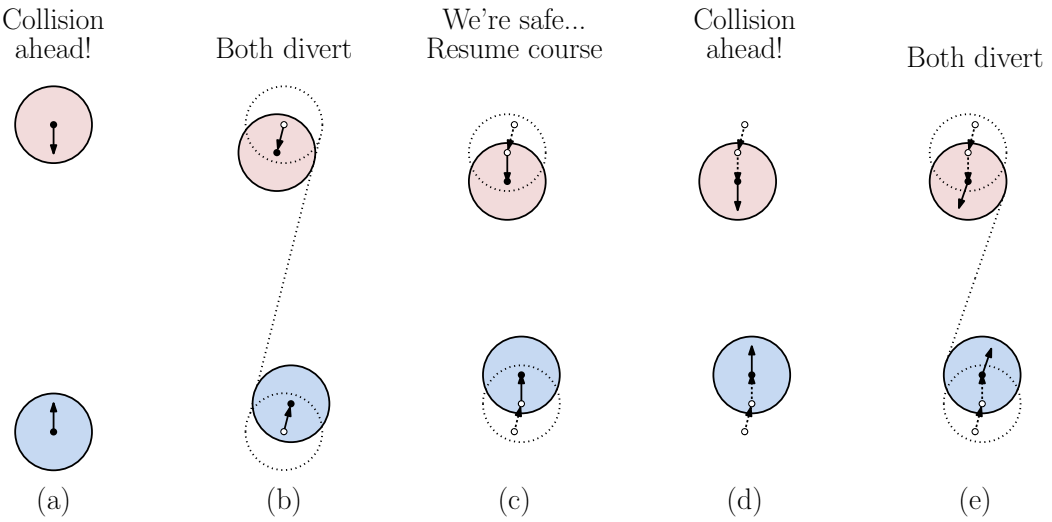

Fig. 4: Oscillation that can result from standard velocity-obstacle motion planning.

Although even humans sometimes engage in this sort of brief oscillation when meeting each other head-on, this sort of repeated oscillation is very unnatural, and is due to the fact that both agents are acting without consideration for what the other agent might reasonable do. The question then is how to fix this?

Reciprocal Velocity Obstacles: The intuition behind fixing the oscillation issue is to share responsibility. We assume that whenever a collision is possible between two agents, both agents perceive the danger and (since they are running the same algorithm) they both know how to avoid it. Rather than having each agent assume total responsibility for correcting its velocity, we instead ask each agent to take on *half* of the responsibility for avoiding the collision. In other words, each agent diverts its path (that is, alters its velocity) by exactly half the amount needed to avoid the collision. It assumes that the other agent will *reciprocate*, by performing the other half of the diversion. It turns out that this greatly reduces the oscillation problem, since two head-on agents will now divert just enough to avoid each other.

This leads to the concept of *reciprocal velocity obstacles*. Before defining this notion, let us recall the sets $\mathrm{CA}^{\tau}_{a|b}(V_b)$, the collision-avoiding velocities for a assuming that b selects its velocity from V_b, and $\mathrm{CA}^{\tau}_{b|a}(V_a)$, the collision-avoiding velocities for b assuming that a selects its velocity from V_a. We say that two sets of candidate velocities V_a and V_b are *reciprocally collision avoiding* if

$$V_a \subseteq \mathrm{CA}^{\tau}_{a|b}(V_b) \qquad \text{and} \qquad V_b \subseteq \mathrm{CA}^{\tau}_{b|a}(V_a).$$

This implies a very harmonious set of candidate velocities, since for any choice $v_a \in V_a$ and $v_b \in V_b$, we can be assured that these two agents will not collide.

Note that there is a complimentary relationship between these two candidate sets. As we increase the possible velocities in V_a, we reduce the possible set of velocities that b can use to avoid a collision, and vice versa. Of course, we would like be as generous as we can, by giving each agent as much flexibility as possible. We say that two such candidate velocity sets are *reciprocally maximal* if

$$V_a \;=\; \mathrm{CA}^{\tau}_{a|b}(V_b) \qquad \text{and} \qquad V_b \;=\; \mathrm{CA}^{\tau}_{b|a}(V_a).$$

Note that we face a tradeoff here, since we could make V_a very large, but at the expense of making V_b very small, and vice versa. There are infinitely many reciprocally maximal collision avoiding sets. So what should guide our search for the best combination of candidate sets? Recall that each agent has its preferred velocity, v_a^* and v_b^*. It would seem natural to generate these sets in a manner that gives each agent the greatest number of options that are close to its preferred velocity. We seek a pair of candidate velocity sets that are *optimal* in the sense that they provide each agent the greatest number of velocities that are close to the agent's preferred velocity.

There are a number of ways of making this concept more formal. Here is one. Consider two pairs (V_a, V_b) and (V'_a, V'_b) of reciprocally maximal collision avoiding sets. For any radius r, $B(v_a^*, r)$ denotes the set of velocities that are within distance r of a's preferred velocity and $B(v_b^*, r)$ denotes the set of velocities that are within distance r of b's preferred velocity. The quantity $\mathrm{area}(V_a \cap B(v_a^*, r))$ can be thought of as the "number" (more accurately the measure) of candidate velocities for a that are close (within distance r) of its preferred velocity. Ideally, we would like both $\mathrm{area}(V_a \cap B(v_a^*, r))$ and $\mathrm{area}(V_b \cap B(v_b^*, r))$ to be large, so that both agents have access to a large number of preferred directions. One way to guarantee that two numbers are large is to guarantee that their minimum is large. Also, we would like the pair (V_a, V_b) to be fair to both agents, in the sense that $\mathrm{area}(V_a \cap B(v_a^*, r)) = \mathrm{area}(V_b \cap B(v_b^*, r))$. This means that they both agents have access to the same "number" of nearby velocities.

Combining the concepts of fairness and maximality, we say that a pair (V_a, V_b) of reciprocally maximal collision avoiding sets is *optimal* if, for all radii $r > 0$, we have

Fair: $\mathrm{area}(V_a \cap B(v_a^*, r)) = \mathrm{area}(V_b \cap B(v_b^*, r))$

Maximal: For any other reciprocal collision avoiding set (V'_a, V'_b),

$$\min(\mathrm{area}(V_a \cap B(v_a^*, r)), \mathrm{area}(V_b \cap B(v_b^*, r))) \;\geq\; \min(\mathrm{area}(V'_a \cap B(v_a^*, r)), \mathrm{area}(V'_b \cap B(v_b^*, r))).$$

Now that we have defined this concept, it is only natural to ask whether we have any hope of computing a pair of sets satisfying such lofty requirements. The remarkable answer is yes, and in fact, it is not that hard to do! The solution is described in a paper by J. van den Berg, M. C. Lin, D. Manocha (see the readings at the start of these notes). They define an *optimal reciprocal collision avoiding pair* of candidate velocities, which they denote by $(\mathrm{ORCA}^{\tau}_{a|b}, \mathrm{ORCA}^{\tau}_{b|a})$, to be a pair of velocity sets that satisfy the above optimality properties.

They show how to compute these two sets as follows. First, let us assume that the preferred velocities of the two agents puts them on a collision course. (This is just for the sake of

illustration. The construction works even if this is not the case.) That is, $v_a^* - v_b^* \in \mathrm{VO}_{a|b}^\tau$. Clearly, we need to divert one agent or both to avoid the collision, and we will like this diversion to be as small as possible. Let u denote the vector on the boundary of $\mathrm{VO}_{a|b}^\tau$ that lies closest to $v_a^* - v_b^*$ (see Fig. 5(a)). Since $\mathrm{VO}_{a|b}^\tau$ is just a truncated cone, it is not too hard to compute the vector u. (There are basically two cases, depending on whether the closest boundary point lies on one of the flat sides or on the circular arc at the base.)

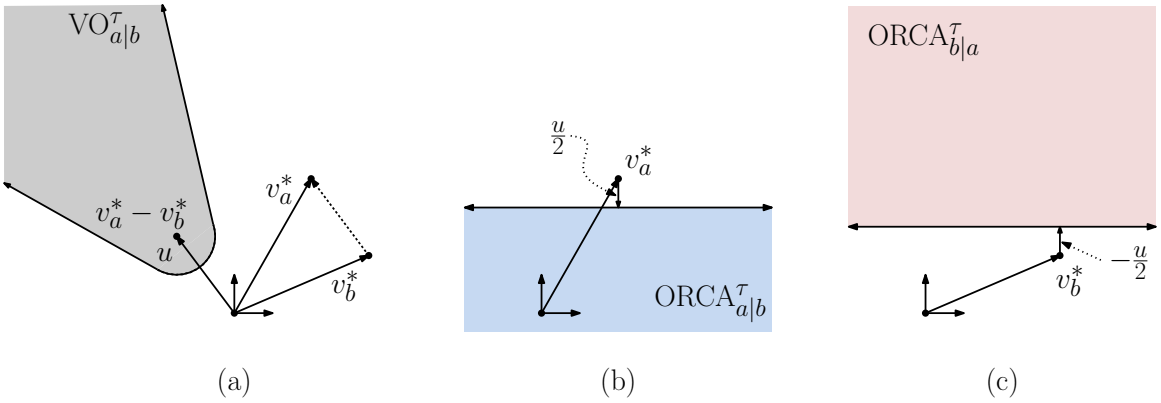

Fig. 5: Computing an optimal reciprocal collision avoiding pair of candidate velocities.

Intuitively, u reflects the amount of relative velocity diversion needed to just barely escape from the collision zone. That is, together a's diversion plus b's diversion (negated) must sum to u. We could split the responsibility however we like to. As we had discussed earlier, for the sake reciprocity, we would prefer that each agent diverts by exactly half of the full amount. That is, a will divert by $u/2$ and b will divert by $-u/2$. (To see why this works, suppose that $v_a' = v_a^* + u/2$ and $v_b' = v_b^* - u/2$. The resulting relative velocity is $v_a' - v_b' = v_a^* - v_b^* + u$, which is a collision-free velocity.)

In general, there are a number of choices that a and b could make to avoid a collision. Let n denote a vector of unit length that points in the same direction as u. We would like a to change its velocity from v_a^* to a velocity whose orthogonal projection onto the vector n is of length at least $\|u/2\|$. The set of allowable diversions defines a half-space (that is, the set of points lying to one side of a line), and is given by the following formula:

$$\mathrm{ORCA}_{a|b}^\tau \;=\; \left\{ v : \left(v - \left(v_a^* + \frac{u}{2} \right) \right) \cdot n \geq 0 \right\},$$

(where the $\cdot$ denotes the dot product of vectors). This formula defines a halfspace that is orthogonal to u and lies at distance $\|u/2\|$ from v_a^* (see Fig. 5(b)). Define $\mathrm{ORCA}_{b|a}^\tau$, symmetrically, but using $-\frac{u}{2}$ instead (see Fig. 5(c)). In their paper, van den Berg, Lin, and Manocha claim that the resulting pair of sets $(\mathrm{ORCA}_{a|b}^\tau, \mathrm{ORCA}_{b|a}^\tau)$ define an optimal reciprocally maximal pair of collision avoiding. In other words, if a selects *any* velocity from $\mathrm{ORCA}_{a|b}^\tau$ and b selects any velocity from $\mathrm{ORCA}_{b|a}^\tau$, and these two sets are both fair and provide the greatest number of velocities that are close to both a and b's ideal velocities.

This suggests a solution to the problem of planning the motion of n bodies. Let $B = \{b_1, \ldots, b_n\}$ denote the set of bodies other than a. Compute the ORCA sets for a relative to

all the other agents in the system. That is, $\bigcap_{i=1}^{n} \text{ORCA}^{\tau}_{a|b_i}$. Since each of these regions is a halfplane, there intersection defines a convex polygon. Next, find the point v'_a in this convex polygon that minimizes the distance to v^*_a. This point defines a's next velocity. By repeating this for every agent in your system, the result is a collection of velocities that are mutually collision free, and are as close as possible to the ideal velocities.

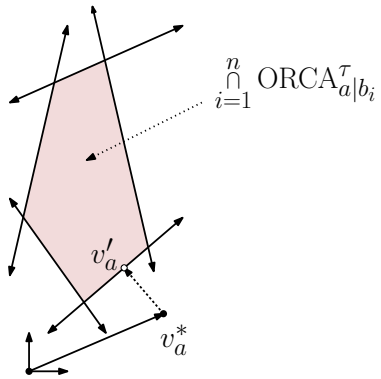

Fig. 6: Computing agent a's velocity.

There are two shortcomings with this approach. First, if the agents are very close to one another, it may be that the intersection of the collision-free regions is empty. In this case, we may need to find an alternate strategy for computing a's velocity (or simply accept the possibility of an intersection).

The other shortcoming is that it requires every agent to know the preferred velocity $v^*_{b_i}$ for each of the other objects in the system. While the simulator may know this, it is not reasonable to assume that every agent knows this information. A reasonable alternative is to form an estimate of the *current velocity*, and use that instead. The theory is that most of the time, objects will tend to move in their preferred direction.