2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Local Reactive Robot Navigation: a Comparison between Reciprocal Velocity Obstacle Variants and Human-Like Behavior

Jerome Guzzi, Alessandro Giusti, Luca M. Gambardella, Gianni A. Di Caro

Abstract— **Most local robot navigation algorithms are based on the concept of velocity obstacle, a mechanistic approach to the navigation problem in which a solution is engineered from scratch. Over the years, a number of different velocity obstacle variants have been developed to effectively handle multi-robot systems. In parallel, an alternative, human-inspired approach for robot navigation has been recently proposed, which derives from the observation and modeling of crowds of pedestrians.**

We discuss similarities and differences among two broadly used obstacle-velocity variants, namely Hybrid Reciprocal Velocity Obstacle and Optimal Reciprocal Collision Avoidance, and the human-inspired approach. How do these differences (which are often subtle) impact performance, and why? We answer these questions through extensive simulation experiments, wherein we evaluate the the algorithms for safety, trajectory efficiency, and emergence of collective behaviors, in different challenging multi-robot scenarios using both ideal and realistic models for robots and sensing.

I. Introduction

Local navigation and collision avoidance is a fundamental task in mobile robotics. The problem is commonly solved using reactive algorithms, which at each control step decide the desired motion towards a target (e.g., a waypoint provided by an higher-level path planning algorithm) accounting for the presence and expected motion of the perceived obstacles.

Early studies dealt with static obstacles sensed by range sensors [1], explicitly considering non-holonomic motion constrains [2], and using decision trees to react to troublesome situations [3]. Among the many proposed techniques to cope with moving obstacles, which include potential fields created around obstacles [4], [5], stochastic dynamic programming [6], and many others (see [7] for an overview), the most commonly adopted ones are based on the concept of *velocity obstacle* [8], i.e., the set of all velocities that will eventually result in a collision. The remaining velocities are safe, and the robot chooses the best one among these.

When many agents implementing the same navigation technique share the same space, velocity obstacle methods may lead to undesirable behaviors, such as oscillations or excessive deviations from the optimal trajectories. Several methods have been proposed to improve the implicit coordination among the agents. For instance, an agent may take only half of the responsibility to avoid an on-course collision, assuming that the other agent will do the same. In this case, one should consider the so-called *reciprocal velocity obstacle* [9] from the set of safe velocities. This enables trajectories that are both smooth and collision free, provided that robots are able to exactly perceive the position and speed of each other, and that some coordination exists, such that the robots choose to pass each other from the same relative side, which prevents oscillatory behaviors.

Two common approaches to enforce the latter condition are: restricting the safe velocity space to half-planes constructed from the tangent spaces of velocity obstacles, truncated by a finite time horizon (Optimal Reciprocal Collision Avoidance, ORCA [10]), or, artificially enlarging one side of the reciprocal velocity obstacle (Hybrid Reciprocal Velocity Obstacle, HRVO [11]). Both techniques can be applied to non-holonomic differential driven robots [12], [13] with localization and sensing uncertainties (an implementation for the Robot Operating System (ROS) is also available [14]). In general, velocity obstacle approaches can take into account specific mobility constraints as well as nonlinear or unpredictable motion of obstacles [15], [16]. Moreover, Lalish and Morgansen [17] recently studied *explicit* coordination techniques for escaping from collision-prone trajectories.

Instead of designing an ad-hoc navigation algorithms from the drawing board, one may wonder how nature has solved the same problem, being local navigation techniques a key feature of human and animal behavior. How do such techniques work? Many studies tried to answer this question, also motivated by the need of creating simulations of human crowds which are accurate (for guiding architectural design of safe public spaces and events) or even just plausible-looking (for animating virtual characters in movies or video games [18]). In particular, pedestrian navigation in crowds has been modeled by a repulsive potential field based on neighbors' positions (*social force* [19]). Improvements of such a model can account for information on neighbors' velocities [20] and integrate the desired velocity given by a reciprocal velocity obstacle planning [21].

An alternative accurate approach for explaining pedestrian behavior was recently proposed by Moussaid et al. [22], where the agent uses a simple heuristic to choose an heading which minimizes the expected distance to the target while avoiding possible collisions. The heuristic is shown to model very accurately the behavior of crowds. In previous work [23], we built on such heuristic with the goal to provide a *human-like* (and, consequently, more human-friendly because of being more predictable) robot navigation technique, which was also shown to yield macroscopic emergent behaviors similar to those observed in human crowds [24].

Main contributions. In this paper, we consider two popular obstacle-velocity variants HRVO [11] and ORCA [10], as well as our human-like approach, HL [23]. First, in Sec-

Authors are with the Dalle Molle Institute for Artificial Intelligence (IDSIA), USI-SUPSI, Manno-Lugano, Switzerland. E-mail: {jerome,alessandrog,luca,gianni}@idsia.ch

tion II, we discuss similarities and differences among their core mechanisms. Then, in Section IV, we report the results of an extensive simulation campaign (amounting to a total of 4 years of simulated time) where the different algorithms are evaluated in various multi-robot navigation scenarios (described in Section III), quantifying both the *safety* and the performance (*efficiency*) aspect of the resulting trajectories. Robot navigation literature has mostly focused on the former aspect, whereas crowd modeling literature was primarily interested on *computational* performance, which is critical when simulating crowds comprising thousands of agents. Trajectory efficiency is rarely investigated quantitatively in either field, to the point that, to the best of our knowledge, no quantitative comparison of different reactive local navigation algorithms has been published before (comparing local navigation methods is an interesting problem in itself, and its inherent challenges are discussed in [25]).

II. Models

In this section we describe the inner workings of the main velocity obstacle algorithm (A-VO), of two reciprocal velocity obstacle variants (A-HRVO and A-ORCA), and of the human-like algorithm (A-HL).

The following 2D reference frame shown in Figure 1 is used for all algorithms. A moving agent a is directed towards a target point $\vec{t}$. We first consider a circular holonomic agent, characterized by radius r_a, position $\vec{p}_a$, current velocity $\vec{v}_a$, and maximal speed $v_a^{\max}$. At each control step, the agent senses the environment and selects a new desired velocity $\vec{v}_a^{\text{des}}$ in order to safely navigate towards $\vec{t}$. In the following, we describe how a reacts when detecting a moving obstacle o with radius r_o, position $\vec{p}_o$, and velocity $\vec{v}_o$. All symbols denote values at the current time.

A. Selection of Desired Velocity with a Single Obstacle

1) Velocity Obstacle (A-VO [8]). The velocity obstacle VO_o is defined as the set of velocities that would lead to a collision with the obstacle o under the assumption that o keeps its current velocity. VO_o is constructed in the velocity space by translating the collision cone C_o (i.e., the set of velocities that eventually lead to a collision if o remains at its current position) by the vector $\vec{v}_o$. In order to avoid collisions, at each control step the agent decides for an optimal velocity *outside* VO_o.

A-VO does not dictate a specific choice of velocity outside of the forbidden set VO_o. Typically, one chooses the velocity within the safe available velocity space (depicted in green in Figure 1) that deviates least from the preferred velocity $\vec{v}_a^{\text{pref}} = v_a^{\max}\frac{\vec{t}-\vec{p}_a}{\|\vec{t}-\vec{p}_a\|}$, that is, the velocity with maximal speed headed straight towards the target:

$$\vec{v}_a^{\text{des}} = \underset{\vec{v}\in\{\vec{v}\in\mathbb{R}^2/\|\vec{v}\|\leqslant v_a^{\max}\}\setminus VO_o}{\arg\min} \| \vec{v} - \vec{v}_a^{\text{pref}} \|. \qquad (1)$$

If no constraints on acceleration are imposed, $\vec{v}_a$ is set to $\vec{v}_a^{\text{des}}$ at the end of the control step.

2) Reciprocal Velocity Obstacle (A-RVO [9]). When the obstacle o is an agent that is also adjusting its velocity, oscillations and unsafe trajectories may occur when using the plain A-VO algorithm, because there is no guarantee that the desired velocity chosen by a remains safe after the concurrent change of velocity by o.

If the agent knows that o is in fact another agent implementing its same behavior, the issue can be addressed by taking *half* of the responsibility to avoid the collision: then, the collision cone C_o is moved by $\frac{1}{2}\vec{v}_o+\frac{1}{2}\vec{v}_a$ instead of by $\vec{v}_o$, yielding to the *reciprocal* velocity obstacle $\text{RVO}_{o|a}$. If both agents choose the desired velocity from the complement of $\text{RVO}_{o|a}$, they will escape the collision course in one control step, provided that both choose to adjust the velocity towards the same (relative) side (i.e., either both are steering left, or both are steering right). Instead, an oscillatory behavior may occur.

3) Hybrid Reciprocal Velocity Obstacle (A-HRVO [11]). In order to force such implicit coordination for steering, the reciprocal velocity obstacle $\text{RVO}_{o|a}$ is asymmetrically enlarged. In particular, if $\vec{v}_a$ is in the left half-plane respect to the bisector of $\text{RVO}_{o|a}$, then the right half of $\text{RVO}_{o|a}$ is substituted with the right half of VO_o, as depicted in Figure 1. If $\vec{v}_a$ lies instead in the right half-plane, the opposite occurs. The resulting forbidden area is named *hybrid reciprocal velocity obstacle*, $\text{HRVO}_{o|a}$. Intuitively, agent a takes half of the responsibility to avoid a collision when choosing to pass to the left, whereas full responsibility is taken if choosing to pass to the right.

The A-HRVO approach leads to safe paths without oscillatory behaviors, even when more than two agents are involved.

4) Optimal Collision Reciprocal Avoidance (A-ORCA [10]). Let $\vec{v}_a^*$ represent an optimal velocity that agent a would like to maintain. In the following, we set $\vec{v}_a^* = \vec{v}_a$, where $\vec{v}_a$ is the current velocity of a (see below). In A-ORCA a finite time horizon τ is considered: beyond τ future collisions are ignored. Consequently, the velocity obstacle VO_o (with apex at $\vec{v}_o^*$) is truncated to VO_o^τ. In practice, this removes the apex of VO_o, which corresponds to the velocities that would lead to a collision after a larger amount of time.

Let $\vec{q}$ be the point on the border of VO_o^τ that is nearest to $\vec{v}_a^*$, and let $\vec{u}$ be the vector connecting $\vec{q}$ to $\vec{v}_a^*$:

$$\vec{q} = \underset{\vec{v}\in\partial\text{VO}_o^\tau}{\arg\min} \| \vec{v} - \vec{v}_a^* \|, \quad \vec{u} = \vec{q} - \vec{v}_a^* ; \qquad (2)$$

also, let $\vec{n}$ be the outwards normal of VO_o^τ at $\vec{q}$. The half-plane $\text{ORCA}_{o|a}$ is defined as the half-plane perpendicular to $\vec{n}$ at point $\vec{v_a}^* + \frac{1}{2}\vec{u}$. Its complement $\overline{\text{ORCA}}_{o|a}$ is the set of forbidden velocities induced by obstacle o on agent a.

Similarly to A-RVO and A-HRVO, the A-ORCA agent takes half of the responsibility to escape from a collision course by adding $\frac{1}{2}\vec{u}$ to its velocity, expecting o to do the same by adding $-\frac{1}{2}\vec{u}$ to its velocity, which would result in collision-free paths. It is additionally possible to formally prove smoothness and safety of the resulting paths. It has also been proven that ORCA half-planes are the largest set that can ensure this result (hence the name *optimal*).

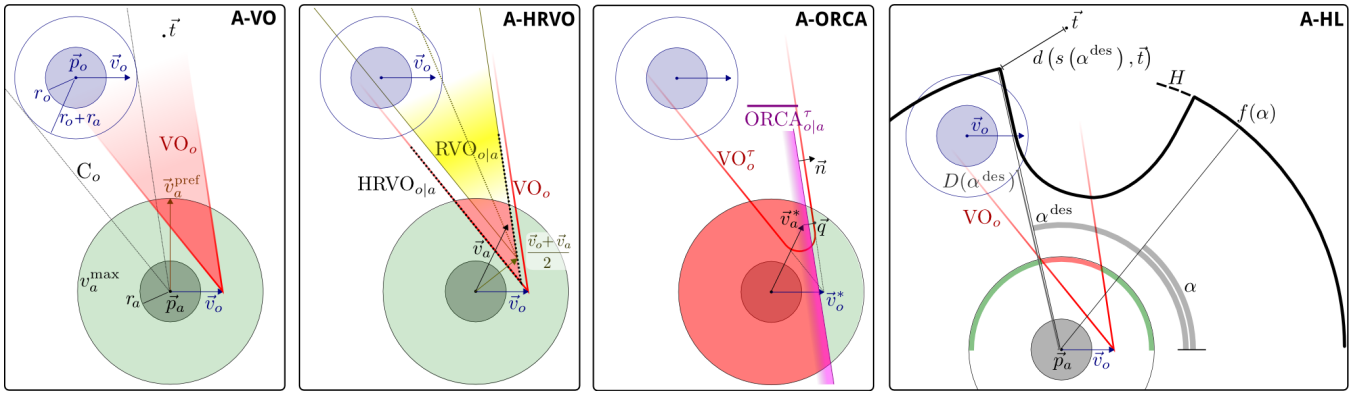

Fig. 1. Illustration of the main entities and notations for A-VO, A-HRVO, A-ORCA, and A-HL.

The time horizon τ represents a critical parameter in A-ORCA, which greatly affects navigation performance in a given environment (in Section III-A we test the effect of different values for τ). Also the choice of which velocity to use as $\vec{v}_a^*$ influences the algorithm behavior. For consistency with other algorithms, hereafter we set $\vec{v}_a^*$ to the current velocity $\vec{v}_a$, which is directly observed by the other agents and thus does not need to be explicitly communicated, and still yields a good behavior [10].

5) Human-Like Navigation (A-HL [23]). The A-HL algorithm works in a significantly different way than the approaches based on velocity obstacles. For a given agent a, the behavior of A-HL is centered on the computation of the function $f_o(\alpha)$. $f_o(\alpha)$ maps an angle α to the free distance that the agent could travel (bounded by a finite horizon H) before colliding with o, under the following assumptions: (i) a moves at its maximum speed $v_a^{\max}$; (ii) o keeps its current velocity (see [26] about how to compute f).

Once $f_o(\alpha)$ is computed, the agent chooses the desired direction α^{des} as follows. Let $\vec{c}_o(\alpha) = f_o(\alpha)\vec{e}(\alpha)$ be the expected point of collision with o, where $\vec{e}(\alpha)$ denotes the unit vector in direction α. Let $s(\alpha)$ denote the segment connecting $\vec{p}_a$ and $\vec{c}_o$, and let $d(\cdot,\cdot)$ denote the distance between a segment and a point. Then:

$$\alpha^{\text{des}} = \underset{\alpha\in[0,2\pi]}{\arg\min}\, d\left(s\left(\alpha\right), \vec{t}\,\right). \tag{3}$$

Intuitively, the agent moves towards the direction that ensures to reach a point as close as possible to the target before colliding with o.

To assure a safe behavior, the agent cautiously selects the desired speed v_a^{des} in order to allow itself to stop in a fixed time η within the free distance $D_o(\alpha^{\text{des}}) \in [0, H]$, as seen in direction α^{des} at the *current time*:

$$v_a{}^{\text{des}} = \min\left(v_a^{\max}, \frac{D_o\left(\alpha^{\text{des}}\right)}{\eta}\right). \tag{4}$$

The algorithm additionally smooths out the velocity profile following the same behaviors as observed in pedestrians and human drivers. In particular, velocity is adjusted over a fixed time interval τ to $\vec{v}_a^{\text{des}} = v_a^{\text{des}}\vec{e}(\alpha^{\text{des}})$, according to an exponential law, $\frac{d\vec{v}}{dt} = \frac{\vec{v}^{\text{des}} - \vec{v}}{\tau}$, which is numerically integrated at each control step. Controlled laboratory experiments measured $\eta = \tau = 0.5$s for pedestrians in normal walking conditions [26].

B. Multiple Obstacles

If multiple obstacles are visible, the extension of all the previous navigation behaviors is straightforward. In A-VO and its relatives, the agent takes the forbidden velocity space as the union of the forbidden set induced by the obstacles

$$\text{FVO} = \bigcup_{o\,\in\,\text{Visible obstacles}} \text{VO}_o \tag{5}$$

and search for an optimal velocity outside of FVO. VO_o is substituted by $\text{RVO}_{o|a}$, $\text{HRVO}_{o|a}$ and $\overline{\text{ORCA}}_{o|a}$ in the corresponding algorithms.

In the case the set of the allowed velocities turns out to be empty, different solutions, with different grades of safety guarantees, are available. In particular, for A-VO, A-RVO, A-HRVO, and A-ORCA, it is possible to:

- remove from the computation of FVO the furthest agents, and repeat it until an allowed velocity is found;
- add a penalty term to the penetration of the FVO and minimize the cost function

$$cost(\vec{v}) = \frac{w}{t_c(\vec{v})} + \parallel \vec{v} - \vec{v}^{\,\text{pref}} \parallel, \tag{6}$$

  where $t_c(\vec{v})$ is the time up to the first upcoming collision when moving at velocity $\vec{v}$, and w is a weight that fixes the tradeoff between performance and safety;
- choose the velocity that minimizes penetration in FVO.

Alternatives only available in A-ORCA consist of:

- reducing the time horizon τ;
- setting $\vec{v}^{\,*} = \vec{0}$ until upcoming collisions are cleared, to guarantee that at least $\vec{v} = \vec{0}$ is a valid choice.

Handling multiple obstacles in A-HL simply requires to account for all visible obstacles when building

$$f(\alpha) = \min_{o\in\text{visible obstacles}} f_o(\alpha), \quad D(\alpha) = \min_{o\in\text{visible obstacles}} D_o(\alpha),$$

then the approach proceeds as described.

Figure 2 shows a configuration in which the paths resulting from the different behaviors are very diverging.

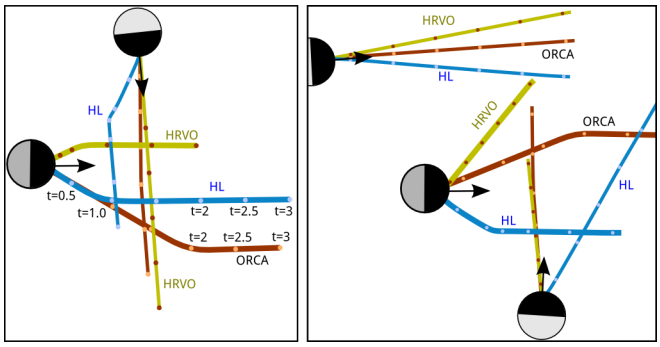

Fig. 2. Illustration of different paths taken by A-HRVO, A-ORCA, and A-HL algorithms given an initial condition and velocity vector for the agents. Each colored line represents the path the agent would take in the following 3 seconds if all robots were driven by the indicated algorithm. Points along the path mark intervals of 0.5 seconds. The initial condition was selected by randomly sampling possible configurations and choosing the one with the highest difference among the paths produced by the three algorithms.

C. Non Holonomic Agents

All algorithms described above assumed an holonomic agent which is capable to immediately adjust its speed in any direction. Instead, we now consider a two wheeled differential driven agent a with wheel axis w_a and current heading α_a. There are various possibilities to adapt the above algorithms accordingly to its mobility constraints.

1) Holonomic Desidered Velocity. A agent may compute $\vec{v}^{\text{des}}$ as if it was holonomic. Then, the desired linear speed is defined as $v^{\text{des}} = ||\vec{v}^{\text{des}}||$ and the angular speed as

$$\omega^{\text{des}} = \frac{\alpha^{\text{des}} - \alpha}{\tau_{\text{rot}}}, \tag{7}$$

so that a rotates in a fixed time τ_{rot} towards the direction α^{des} of $\vec{v}^{\text{des}}$. Finally, one translates linear and angular speeds into left and right wheel speeds $w_L^{\text{des}} = v^{\text{des}} - \frac{w}{2}\omega^{\text{des}}$, $w_R^{\text{des}} = v^{\text{des}} + \frac{w}{2}\omega^{\text{des}}$, limited to the maximal speed of wheels.

2) Effective Center. Following [12], the second possibility is to consider the agents as being contained into an effective circle with a forward-shifted center $\pi_a = p_a + \rho_a \vec{e}(\alpha_a)$ and radius $r_a + \rho_a$. The advantage of this transformation is that there is a invertible map T between wheel speeds and velocities of the effective center, such that the agent is able follow an arbitrary path of its effective center like it were holonomic. In the following we fix $\rho_a = \frac{w_a}{2}$ so that the transformation is $T(w_L, w_R) = \dot{\vec{\pi}} = ||v_a||\vec{e}(\alpha) + \rho_a \dot{\alpha}\vec{e}_\perp(\alpha) = \frac{w_L+w_R}{2}\vec{e}(\alpha) + \frac{w_R-w_L}{2}\vec{e}_\perp(\alpha)$, where $\vec{e}$ (pointing forwards) and $\vec{e}_\perp$ form an orthonormal frame attached to the robot. One computes $\vec{v}^{\text{des}}$ for the effective circle and transforms it back to wheel speeds with

$$w_L^{\text{des}} = \vec{v}^{\text{des}} \cdot (\vec{e}(\alpha) - \vec{e}_\perp(\alpha)), \; w_R^{\text{des}} = \vec{v}^{\text{des}} \cdot (\vec{e}(\alpha) + \vec{e}_\perp(\alpha)).$$

The maximal speed that the effective center is able to reach in all directions is $v_{\text{eff}}^{\max} = \frac{v_a^{\max}}{\sqrt{2}}$. In turn, this reduces the maximal selected wheel speed to $v_{\text{eff}}^{\max}$ too.

Other possibilities include the addition of constraints that reduce the admissible velocity space over velocities that are actually reachable by the agent before the next control step.

D. Safety Guarantees

Let us assume that agents are equipped with perfect sensors, and that they adjust their desired velocity once every *finite* time step Δt. In order to prevent collisions, one can add a safety margin m to the physical radius of the obstacles, then artificially force that agents that enter into the safety margin of an obstacle o to nullify the components of $\vec{v}_a^{\text{des}}$ which point towards o. This is exactly how safety is enforced in A-HL, where $f_o(\alpha) = D_o(\alpha) = 0$ for all α that point towards o when the distance between a and o is less than m.

For an agent with no constraints on acceleration, moving together with agents with the same upper bound on speed $v^{\max}$, collision-free behavior is ensured if $m > M = 2v^{\max}\Delta t$. If the agents employ the A-HL behavior with speed modulated by $\tau > 0$, the bound is instead $M = 2v^{\max}(\Delta t + \tau)$, because the agents need an additional amount of space to come to a complete stop. Note that M represents an *upper bound* on the minimal safety margin required for collision-free behavior. In practice, much shorter safety margins can be safely adopted.

Non-holonomic agents demand extra care when their selection of desired velocity does not take the motion constraints into account (Section II-C.1) because they need additional space to turn towards the desired heading. In the worst case scenario, when two facing robots moving at full speed do an half turn, the space needed between them when moving accordingly to Equation (7) with $\dot{\alpha} = \omega^{\text{des}}$ and $\alpha^{\text{des}} = \pi$ is given by $2\Delta x = 2\int_0^{t(\frac{\pi}{2})} v_x dt = 2\int_0^{\frac{\pi}{2}} v^{\max}\cos(\alpha)\frac{d\alpha}{\dot{\alpha}} = 2v^{\max}\tau_{\text{rot}}\int_0^{\frac{\pi}{2}} \frac{\cos(\alpha)}{\pi-\alpha} d\alpha \cong 0.8v^{\max}\tau_{\text{rot}}$. This needs to be added to M to ensure safety. If the controller is based on A-HL with $\tau > 0$, is is necessary to add another finite term proportional to τ. The point here is that it is always possible to ensure safety by a large enough safety margin provided that the agents follow the rule above. The tradeoff is that a larger safety margin generally leads to worse performance because it reduces the available free space.

When sensing is imperfect due to inaccuracies or a limited field of view, the above guarantees do not hold. Then, one should search for an optimal tradeoff between safety and performance (like pedestrians and drivers unconsciously do). We investigate such tradeoff in Section III-A.

E. Comparisons

All algorithms we considered above have a common trait: they anticipate future collisions by using the current sensing information for position, velocity, and shape of the agent and of surrounding obstacles[1]. All use a linear prediction of the obstacles' trajectories to compute a time-to-collision estimate, then select the velocity that minimizes their deviation from the straight line towards the target.

A-RVO, A-HRVO and A-ORCA explicitly share the collision avoidance responsibility amongst agents, which leads to improved performance. A-HL indirectly obtains the same

[1]In contrast, methods based on potential fields (also known in sociology as *social forces*) do not explicitly perform this sort of extrapolation.

effect by modulating velocities smoothly using the τ parameter: an agent, while smoothly turning to avoid others, has sufficient time to acknowledge the obstacles' actions.

All considered algorithms only use currently sensed information and bear no history or state information, i.e. they are purely *reactive* and *stateless*[2]; at the same time, all algorithms *proactively* avoid collisions and anticipate the motion of others.

The most prominent peculiarity of A-HL is that it performs a one-dimensional search over the direction of the desired velocity, choosing the one that minimize the *spatial* distance to the target. A-VO and all derivatives, instead, perform a search over (a subset of) the two-dimensional velocity space: then the desired speed is chosen in order to minimize the *velocity-space* distance to the optimal velocity – i.e. the velocity directed towards the target with maximal speed. Note that because A-HL acts to minimize space distance, in no circumstances it will dictate to move farther away from the target. Instead, A-ORCA and A-HRVO may exhibit such behavior, when the forward half of the velocity space is forbidden (i.e., when moving backwards is the only solution to avoid a future collision).

Another peculiarity of the A-HL algorithm is that it does not explicitly exclude directions that could lead to future collisions: such directions are just penalized in the search of the desired direction. Instead, A-VO and its derivations irreversibly forbid all velocities leading to a collision, unless forced by the absence of alternatives (i.e., an empty set of safe velocities, as discussed in Section II-B). In other words, A-VO and its variants start by searching for the set of safe velocities, then select the one that maximizes performance, whereas A-HL greedily selects a direction maximizing performance (accounting for obstacles), and only as a second step it accordingly adjusts the speed in order to ensure safety.

When no safe velocity is available, A-RVO minimizes the cost function of Equation (6)

$$\text{cost}_{\text{RVO}}(\alpha, v) = \frac{w}{g(\alpha, v)} \frac{v}{v^{\max}} + \left\| \vec{e}^{\,\text{pref}} - \vec{e}(\alpha) \frac{v}{v^{\max}} \right\|, \quad (8)$$

where $g(\alpha, v) = t_c(\vec{v}) \cdot v$ indicates the distance that the agent could travel before any collisions in direction α when moving at an arbitrary speed v. This bears some resemblance to, but differs significantly from, the cost function optimized by A-HL (with a target at distance d_t in Equation (3)):

$$\text{cost}_{\text{HL}}(\alpha) = \left\| \vec{e}^{\,\text{pref}} - \vec{e}(\alpha) \frac{g(\alpha, v^{\text{opt}})}{d_t} \right\|. \quad (9)$$

III. Experimental Setup

Simulation experiments are performed with a custom simulator including an accurate physical simulation of the environment and a model for sensing inaccuracies. Robot navigation controllers are evaluated at a frequency of 10Hz. The system was used to run a large batch of simulated experiments in parallel on a 180-nodes cluster: data reported below are equivalent to a total of 4 years of simulated time.

[2]This does not necessarily apply to the sensing subsystems. For instance, history of obstacles' positions could be maintained in order to determine their speed.

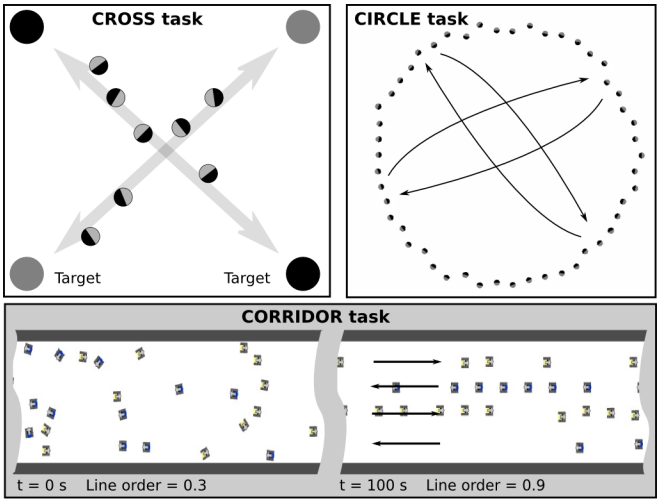

Fig. 3. The three navigation tasks considered in the simulations.

A. Simulation Scenarios

We consider three navigation *tasks*, illustrated in Figure 3:

- *Cross*, in which agents need to travel back and forth between two targets located at the opposite vertices of a square with an edge of 4 meters, This creates a crossroad in the middle where robots frequently need to adjust their trajectories in order to avoid collisions. It provides a realistic testing scenario for robots.
- *Circle*, in which agents initially placed at regular intervals along a circumference of radius 5 meters need to exchange the position with the agent located at the opposite point with respect to the center. It is a commonly used scenario in A-VO studies and it provides a sort of benchmark scenario.
- *Infinite corridor*, in which two groups of agents travel towards opposite directions in a corridor of width w, varied from 0.5 to 4 meters in the experiments. We simulate a finite-length section of the corridor, of length $l = 16$m. Its two ends "wrap around" and connect with each other, as they were the lateral surface of a cylinder. This is a setup which is commonly considered in crowd analysis literature [26].

Simulations are run using two different *robot models*: an ideal *holonomic* robot, and a small, *differential-driven* (non-holonomic) robot with the same dynamical characteristics as the *foot-bot* robot platform [27]. Both models share the same physical characteristics: radius $r = 8.5$ cm, wheel axis 13 cm, $v^{\max} = 30$ cm/s. Up to a scale factor, the foot-bot robot also well represents the characteristics of other common differential-driven robots.

In real robot implementations, perception of the environment (i.e., positions of navigation targets and of other robots) is commonly affected by major sensing inaccuracies, which in turn affect navigation performance and safety.

To address this issue, in our simulations we consider two different sensing models. A *perfect sensing* model, in which all robots within an assiged range are perfectly detected, and a *realistic*, camera-based sensing model, in which neighbors are only perceived when not occluded and within a given angular field of view (centered on the direction the robot is currently facing). Moreover, neighbor localization and speed estimation is affected by a fixed angular (bearing) error and a depth (range) error proportional to the neighbor's distance.

In previous work [23], the adopted simulation and sensing models were validated and compared to a real implementation on a swarm of fully-autonomous foot-bot robots. The robots solved a local navigation task using the on-board A-HL algorithm using a sensing subsystem based on a low-resolution embedded camera.

B. Algorithms and parameters

Simulated agents are provided with sensing information according to the considered sensing model (perfect or realistic). At each control step, at a rate of 10Hz, an agent feeds this information to the selected algorithm, and issue speed commands to drive the simulated wheel actuators. Physics' simulation runs at an higher rate and yields a realistic rendering of robot dynamics.

For comparisons, We consider four different navigation algorithms For holonomic robots we used ORCA, HRVO and HL, which directly controls the movements of the robots. Non holonomic robots are controlled with ORCA, HRVO and HL as described in Section II-C.1, by translating the holonomic desired velocity into wheel speeds. In addition, we also consider ORCA-NH, which explicitly takes into account the non-holonomicity of the robot when computing the desired wheel velocities, as discussed in Section II-C.2.

1) HL. We use the implementation described in [23] based on the model of Section II-A.5, with parameters $\tau = 0.125$s and $\eta = 0.5$s, which yield a reactive but at the same time smooth behavior. The desired angle is determined by splitting the field of view in 100 angular steps.

2) HRVO. We use the implementation provided by the C++ *HRVO Library* [28], which is based on the model described in Section II-A.3. The desired velocity is found in the velocity space through a linear optimization technique. The implementation does not have free parameters.

3) ORCA. We use the implementation provided by the C++ *RVO2 Library* library [29], which is based on the model described in Section II-A.4. The desired velocity is found in the velocity space through a linear optimization technique. The time horizon H is a free parameter which we investigate in a specific experiment below. In other experiments, we report several values for H, selected among those maximizing performance. A large time horizon allows the robot to anticipate crowding and avoid congestion, but at the same time penalizes it with a reduction of speed and a longer, more conservative path.

4) ORCA-NH. We use the same controller as ORCA, but apply it to the effective center and effective radius (see Section II-C.2) of the non-holonomic agents. Note that, unlike in [12], the same geometric transformation is not applied to perceived obstacles that are non-holonomic agents.

C. Performance Metrics

For each scenario and navigation algorithm, we compute the following performance measures:

- *Safety* of the swarm, measured by the number of collisions per hour per agent.
- *Relative throughput* of the swarm, which indicates the efficiency in navigating towards the targets. This measure is defined for the *cross* task as the total amount of targets that the robots were able to reach, divided by the amount of targets that the robots could reach in the same time while traveling in straight lines (i.e., ignoring any collision). In the *circle* task, throughput is defined as the minimal time it would take for one robot to reach the opposite side (when traveling in a straight line) divided by the actual time it took, and averaged over all robots. In the *corridor* task, the relative throughput is given by the average speed directed towards the target divided by the maximal admitted speed of the robot. Relative throughput for NH-ORCA takes into account the lower maximal speed that a robot is allowed to achieve, in such a way that it is not penalized.

IV. Experimental Results and Discussion

We initially investigate the safety of the different algorithms and its relation to throughput in relation to different choices for the safety margin (Section IV-A). Then, we discuss the throughput of the different algorithms (Section IV-B). Finally, we discuss the emergence of collective behaviors (Section IV-C). The results of all simulations experiments are reported in Figure 4.

A. Safety

Figure 4a shows the amount of violation of a defined safety margin of $m = 20$cm. The figure reports the probability that at least one obstacle penetrates into this margin by more than a given amount (x axis). This information can guide the choice for a safe and efficient value of m. HL and ORCA result to be significantly safer than HRVO: for these algorithms a margin of $m = 6$cm is found to be sufficient. In Figure 4b, similar results are shown when using non holonomic robots. In this case, plain ORCA becomes less safer and occasionally violates safety margins by more than 10cm, while the ORCA-NH variant proves to be much safer.

Choosing a larger value for m not only improves safety, but it also hinders throughput: the resulting tradeoff is investigated in Figure 4c, using foot-bots with realistic sensing and limited field of view. The HL algorithm performs well by yielding safe operation (less than 10 collisions per hour) and large throughput for any choice of the safety margin. In contrast, safety of ORCA strongly depends on m's choice.

In all the simulations experiments that follow, m was set to 6cm.

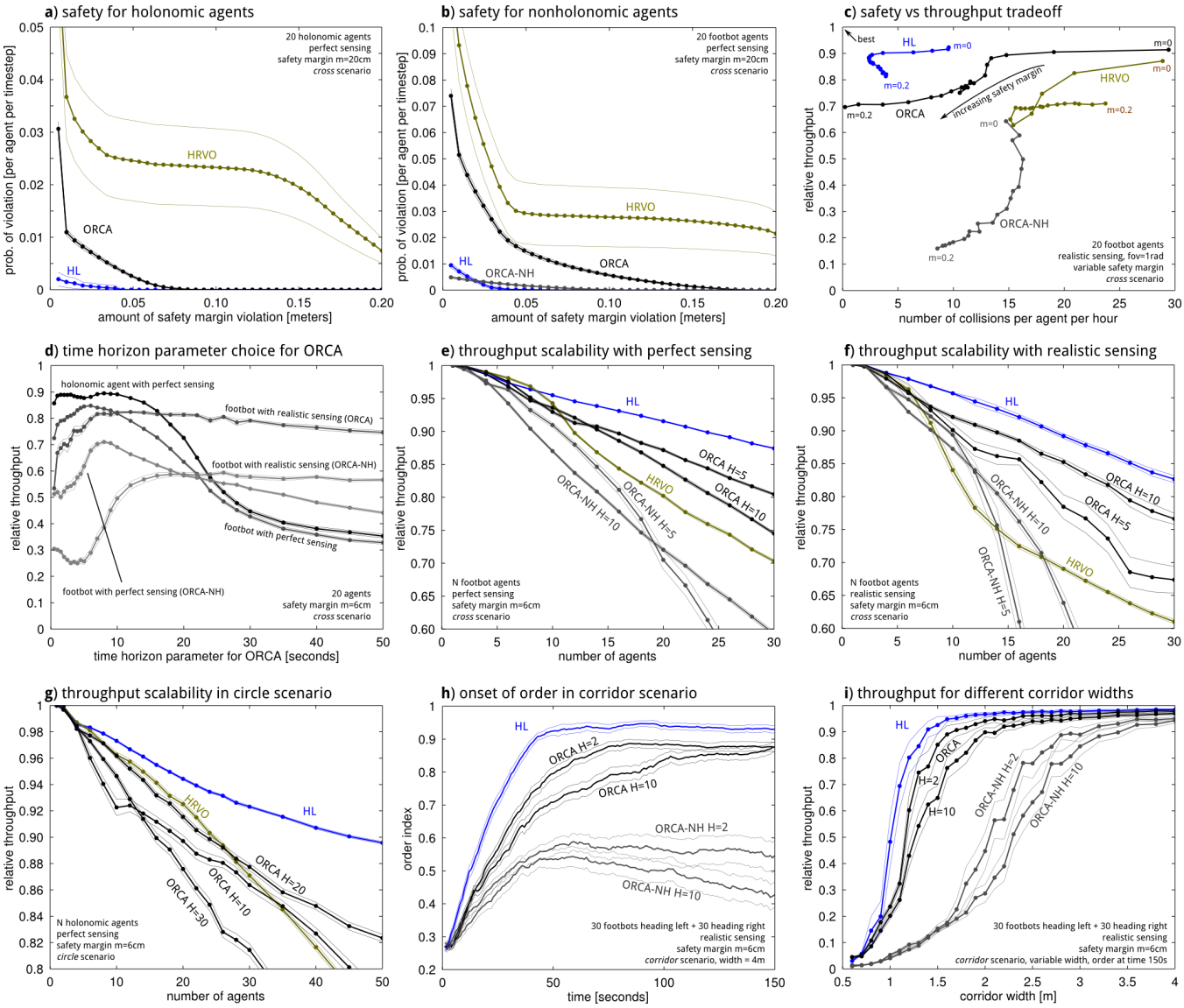

Fig. 4. Simulation results (see text). Thin lines above and below each plot represent 95% confidence intervals.

B. Throughput

Figure 4d explores how the time horizon parameter H of the ORCA algorithm affects the performance of robots with different sensing ability. Although such robots react differently to different values of H, values from 5 to 15 seconds yield good performance in most cases. This information guided our choice for H in the subsequent simulations.

Interestingly, robots with perfect sensing suffer from a time horizon larger than 15s, since they become overly cautious as they account for the trajectories of their peers. In comparison, robots with realistic sensing tend to perform much better in the same conditions, since most peers are invisible because of occlusions, which locally allows for more direct trajectories.

Figure 4e shows how different algorithms cope with increasingly crowded scenarios. As expected, all algorithms reduce their performance when more robots are used, as longer and more complicated trajectories are required to navigate around others and avoid collisions. HL outperforms other algorithms in these scenarios, especially when a relatively large number of robots is considered. This can be explained by the greedy characteristic of HL (see Section II-E).

Figure 4f reports the same results when using realistic instead of perfect sensing. Results do not change significantly, however, now ORCA with $H = 10$ consistently outperforms ORCA with $H = 5$, whereas with perfect sensing the opposite was true. This can be explained by the fact that a large time horizon leads to a more preemptive and cautious behavior, which is an advantage in case of unreliable sensing, and a disadvantage otherwise.

C. Emerging Behaviors

Figure 4g reports the throughput figures obtained when solving the *circle* task, which is especially challenging since the center of the circle becomes very crowded. In this case,

HL has been shown [24] to yield an interesting emerging behavior in which, despite the absence of any explicit coordination, all agents pass on the same relative side of the center, thus creating a sort of "whirlpool" which turns out to be a very efficient solution to the problem. HRVO, which includes an intrinsic symmetry-breaking mechanism, yields the same behavior, and in this scenario enjoys a much better throughput compared to all ORCA-based algorithms.

When two groups of agents navigate through a corridor in opposite directions, emergent behaviors have also been observed both in humans [26] and robots implementing HL [24]: agents tend to organize in ordered lines, which minimizes the need to avoid others [26]. Figure 4h puts the algorithms in the same situation, and quantifies how fast these ordered formations are reached by computing an order index as defined in [26], which ranges from 0 (random distribution of agents going in either directions) to 1 (agents traveling in the same direction are clustered in lines). We observe that both HL and ORCA reach an ordered configuration within 40-50 seconds, but according to significantly different dynamics. Figure 4i considers the same scenario, but reports how throughput changes as the corridor is narrowed. Both HL and ORCA robots manage to efficiently negotiate corridors as narrow as 1.5 meters. In contrast, ORCA-NH, owing to their inability to reach an ordered configuration, yield very low throughput when the corridor is less than 2 meters wide.

V. Conclusions and Ongoing Work

We considered three local navigation techniques for mobile robots, namely Hybrid Reciprocal Velocity Obstacle (HRVO), Optimal Reciprocal Collision Avoidance (ORCA) and a human-like algorithm (HL), and we discussed their internals and highlighted differences and similarities. A simulation study considering several challenging settings and based on both ideal and realistic robot models has shown that the different techniques yield widely variable performance, with HL often outperforming the others.

Ongoing work is focused on extending this comparative study by considering additional path quality metrics. For instance, when robots share spaces with human users, one should aim at generating *friendly* and *acceptable* trajectories. In this case, smoothness and predictability may be more significant measures of performance than throughput.

References

[1] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

[2] 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.

[3] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, 2004.

[4] S. Ge and Y. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots*, pp. 207–222, 2002.

[5] S. Sugiyama, J. Yamada, and T. Yoshikawa, "Path planning of a mobile robot for avoiding moving obstacles with improved velocity control by using the hydrodynamic potential," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010, pp. 1421–1426.

[6] N. Du Toit and J. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transactions on Robotics*, vol. 28, pp. 101–115, 2012.

[7] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

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

[9] J. van den Berg, D. Manocha, and M. Lin, "Reciprocal Velocity Obstacles for real-time multi-agent navigation," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2008, pp. 1928–1935.

[10] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research - Proceedings of the 14th International Symposium ISRR*, ser. Springer Tracts in Advanced Robotics, 2011, vol. 70, pp. 3–19.

[11] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[12] ——, "Smooth and collision-free navigation for multiple robots under differential-drive constraints," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010, pp. 4584–4589.

[13] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Proc. of the Int. Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2010, pp. 1–14.

[14] D. Hennes, D. Claes, W. Meeussen, and K. Tuyls, "Multi-robot collision avoidance with localization uncertainty," in *Proc. of the Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012, pp. 4–8.

[15] J. van den Berg and M. Overmars, "Planning time-minimal safe paths amidst unpredictably moving obstacles," *The Int. Journal of Robotics Research*, vol. 27, pp. 1274–1294, 2008.

[16] A. Wu and J. P. How, "Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles," *Autonomous Robots*, vol. 32, no. 3, pp. 227–242, 2012.

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

[18] S. J. Guy, M. C. Lin, and D. Manocha, "Modeling collision avoidance behavior for virtual humans," in *Proc. of Int. Conf. on Autonomous Agents and Multiagent Sys. (AAMAS)*, 2010, pp. 575–582.

[19] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, pp. 487–490, 2000.

[20] F. Zanlungo, T. Ikeda, and T. Kanda, "Social force model with explicit collision prediction," *EPL (Europhysics Letters)*, vol. 93, no. 6, 2011.

[21] G. Lämmel and M. Plaue, "Getting out of the way: Collision avoiding pedestrian models compared to the real world," *Pedestrian and Evacuation Dynamics*, pp. 1–15, 2012.

[22] M. Moussaïd, D. Helbing, and G. Theraulaz, "How simple rules determine pedestrian behavior and crowd disasters," *Proc. of the National Academy of Sciences of the United States of America*, no. 17, pp. 6884–6888, 2011.

[23] J. Guzzi, A. Giusti, L. M. Gambardella, G. Theraulaz, and G. A. Di Caro, "Human-friendly robot navigation in dynamic environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013, pp. 423–430.

[24] J. Guzzi, A. Giusti, L. M. Gambardella, and G. A. Di Caro, "Bioinspired obstacle avoidance algorithms for robot swarms," in *Proc. of the 7th Int. Conf. on Bio-Inspired Models of Network, Information, and Computing Systems (BIONETICS)*, 2012.

[25] I. Ranó and J. Minguez, "Steps towards the automatic evaluation of robot obstacle avoidance algorithms," in *Workshop on Benchmarking in Robotics, in the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 2006.

[26] M. Moussaïd, D. Helbing, S. Garnier, A. Johansson, M. Combe, and G. Theraulaz, in *Proc. of Royal Society B: Biological Sciences*.

[27] M. Bonani, V. Longchamp, S. Magnenat, P. Rétornaz, D. Burnier, G. Roulet, F. Vaussard, H. Bleuler, and F. Mondada, "The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010, pp. 4187–4193.

[28] "HRVO Library," http://gamma.cs.unc.edu/HRVO.

[29] "RVO2 Library," http://gamma.cs.unc.edu/RVO2.