C-Nav: Distributed Coordination in Crowded Multi-Agent Navigation

Julio Godoy*

Universidad de Concepcion, Chile

Stephen J. Guy, Maria Gini

University of Minnesota, USA

Ioannis Karamouzas

Clemson University, USA

Abstract

In crowded multi-agent navigation, the motion of the agents is significantly constrained by the motion of the nearby agents. This makes planning paths very difficult and leads to inefficient global motion. To address this problem, we propose a distributed approach, which we call *C-Nav*, that introduces politeness into multi agent navigation. With our approach, agents take into account the velocities and goals of their neighbors and optimize their motion accordingly and in real-time. Further, we perform a theoretical analysis of the algorithm, and experimentally demonstrate its advantages in simulation, with hundreds of agents in a variety of scenarios, and in real world navigation tasks with several mobile robots.

Keywords: Multi-agent navigation, multi-agent coordination, robotics

1. Introduction

Decentralized goal-directed navigation of multiple agents in crowded environments has application in a variety of domains such as swarm robotics, traffic engineering, and planning for evacuation. This problem is challenging due to the conflicting constraints induced by the other moving agents; as agents plan paths in a decentralized manner, they often need to recompute their paths in real-time to avoid colliding with the other agents and static obstacles. The problem becomes even harder when temporal constraints are added such as

*Corresponding author

Email addresses: juliogodoy@udec.cl (Julio Godoy), sjguy@umn.edu (Stephen J. Guy), gini@umn.edu (Maria Gini), ioannis@clemson.edu (Ioannis Karamouzas)

when the agents need to reach their destinations in a timely manner while still guaranteeing a collision-free motion. Further, the lack of knowledge of each others' internal states prevents the agents from coordinating their local motion adding another layer of complexity to the decentralized multi-agent navigation problem.

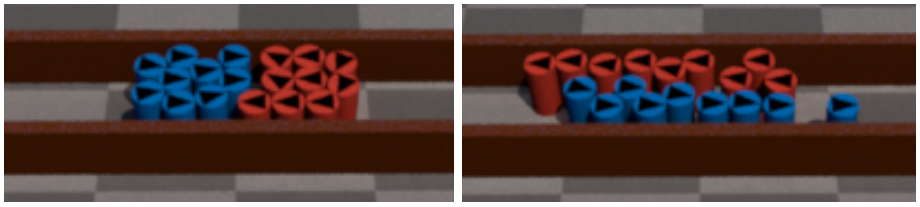
A variety of approaches have been proposed to address this problem, including rule-based techniques [58, 59], geometric solutions [3, 20], and graph-based techniques [40, 64]. Recently, multi-agent navigation methods have been enhanced with machine learning techniques, such as deep learning, allowing agents to learn navigation strategies in a variety of domains. However, even though deep learning techniques have made significant strides towards fully autonomous robot navigation, they still cannot generalize well to unseen domains or handle highly crowded scenarios. Our goal is to have mobile robots that learn in an online and decentralized setting and are robust to new situations while avoiding collisions. Velocity-based approaches, such as ORCA [3], offer a preferred family of methods for multi-agent navigation due to their robustness and their ability to provide collision-free guarantees for the agents' motions. Such approaches allow each agent to directly choose a new collision-free velocity at each cycle of a continuous sensing-acting loop. However, in crowded environments, velocities that are locally optimal for one agent are not necessarily optimal for the entire group of agents. This lack of "social awareness" can result in globally inefficient behavior, which translates into long travel times and an excess of energy expended by the agents.

Ideally, to obtain globally efficient motions and reduce the overall travel time of the agents, a centralized entity needs to compute the best velocity for each agent. However, in a decentralized domain, such central entity does not exist. In this type of domains, agents can only use their limited knowledge of the environment, obtained through local sensing, to compute their motions. This might be enough for agents to avoid colliding with each other, but prevents them from coordinating their motions at the higher level needed to facilitate time- and energy-efficient navigation for the entire system of agents.

Consider, for example, the two group of agents in Figure 1(a). Here, the two groups of agents try to move past each other in a narrow hallway. The agents navigate using a predictive collision-avoidance technique, ORCA, but still end up getting stuck in a congested area. This uncoordinated behavior can be observed, in general, when a dense group of agents with conflicting goals navigate in a constrained environment, and is an example of the "faster is slower" effect [26, 52, 24] studied in the pedestrian navigation literature. In the same domain, it has been observed that when (at least part of) the crowd behaves in a "cooperative" manner, i.e. showing polite behavior towards others, the travel time of the entire crowd is reduced [14, 68, 47].

In this paper, we hypothesize that if agents in multi-robot navigation tasks could behave politely towards others, for example, by choosing velocities that not only benefit themselves but also their neighboring agents, then the efficiency of the global navigation would significantly improve. As such, we seek to develop a navigation method that encourages coordination to emerge through agents'

polite interactions. We accomplish this by allowing agents to account for their neighbors' intended velocities during their motion planning without losing the collision-free guarantees of existing multi-agent navigation frameworks.



(a) ORCA (b) *C-Nav*

Figure 1: Two groups of agents cross paths while moving to the opposite side of a narrow corridor. (a) ORCA agents get stuck in the middle. (b) Using our *C-Nav* approach, agents create lanes in a decentralized manner and move to their goals faster.

To this end, we propose *C-Nav* (short for Coordinated Navigation), a distributed coordination approach that introduces the concept of politeness into a multi-agent navigation framework. Figure 1(b) shows an example of such coordinated motion achieved with our proposed approach. With *C-Nav*, robotic agents improve their global motion in crowded environments by coordinating their local motions, in a distributed fashion, while keeping the collision-free properties of the underlying local navigation framework. This coordination is achieved by the agents using observations of the nearby agents' motion patterns and a limited one-way communication. This low-communication requirement allows *C-Nav* to scale and simulate hundreds of agents in real time. With our approach, agents choose velocities that help their nearby agents to move to their goals, effectively improving the time-efficiency of the entire crowd, as shown in a variety of scenarios in simulation. Further, we demonstrate how accounting for politeness translates into the multi-robot navigation problem, through experiments with physical robots with non-holonomic constraints.

In this work, we assume that all agents move according to the same underlying collision avoidance algorithm. Further, we propose a one-way communication model, without requiring any type of exchange of information or negotiation protocols. Accordingly, we label *C-Nav* as a distributed coordination method, as each agent decides on its behavior autonomously. The type of communication that *C-Nav* employs could be found in communicative robots, such as automated warehouses, future automated vehicles and, in general, in environments where point-to-point communication is not always possible, or when real time constraints prevent more complex negotiation between the agents.

Main Contributions. This paper makes four main contributions. First, we propose a framework that introduces the concept of politeness into multi-agent navigation tasks. Specifically, we show that accounting for nearby agents when selecting an optimal velocity promotes distributed coordination between agents.

Second, we discuss theoretical properties of our proposed method, showing the conditions under which *C-Nav* agents can avoid livelocks. Third, we empirically evaluate *C-Nav* in simulation, in a variety of scenarios, and show that it leads to more efficient global navigation, reducing the travel time as well as the energy expended of all agents as compared to ORCA. Fourth, we evaluate *C-Nav* in multi-robot experiments with three Turtlebot robots navigating in constrained conditions.

This work is an extended version of [17], which introduced the *C-Nav* method for distributed coordination in multi-agent navigation problems. Compared to [17], here we perform a more thorough evaluation of *C-Nav*, highlight the theoretical properties of the method, and provide new, extended experimental analysis including experiments with real robots.

The rest of the paper is organized as follows. In Section 2, we review relevant related work. In Section 3, we formulate the problem of achieving coordinated motions to minimize the travel time of the agents. The *C-Nav* approach and its main insights are introduced in Section 4. In Section 5, we present theoretical properties of our proposed method. Section 6 introduces the empirical evaluation on *C-Nav* with detailed simulation results. The analysis of the method is presented in Section 7. Multi-robot experiments and results are presented in Section 8. Limitations of *C-Nav* are described in Section 9. We conclude and provide directions for future work in Section 10.

2. Related Work

Multi-Agent Navigation. A number of models have been proposed to simulate the motion of agents. In this work, we focus on decentralized multi-agent navigation where each agent plans independently. Such approaches can be traced back to the seminal work of Reynolds on *boids* [58], after which many agent-based approaches have been introduced, including social forces [27], models that account for groups [2], cognitive and behavioral rules [13, 61], biomechanical principles [22] and sociological or psychological factors [54, 21, 57]. However, the majority of such agent-based techniques does not account for the velocities of individual agents which leads to unrealistic behaviors such as oscillations. These problems tend to be exacerbated in densely packed, crowded environments.

To address these issues, *velocity-based* algorithms [11] have been proposed that compute collision-free velocities for the agents using either sampling [50, 34] or optimization-based techniques [3, 23]. In particular, the Optimal Reciprocal Collision Avoidance navigation framework, ORCA [3], plans provably collision-free velocities for the agents and has been successfully applied to simulate high-density crowds [6]. However, ORCA and its variants are not sufficient on their own to generate time-efficient agent behaviors, as computing locally optimal velocities does not always lead to globally efficient motions. As such, we build on the ORCA framework while allowing agents to coordinate their motions, in a distributed manner, in order to improve the global time-efficiency of the crowd.

Multi-Robot Navigation. When translating multi-agent navigation approaches to the physical domain, many challenges arise. For example, the

collision avoidance methods that work in simulation might not be directly applicable with the robots' imperfect sensors and actuators. However, having multiple robots moving in a shared environment means that avoiding collisions with each other is of top priority. Potential fields were among the first methods proposed for collision avoidance in robot navigation [35, 60, 45]. The approach in [62] extends single agent collision avoidance methods for multiple robots, decoupling path planning and coordination. A review of algorithms for collision-free navigation of mobile robots is presented in [31], including different models of sensors and robot kinematics, as well as different assumptions about the environment.

More recently, Alonso-Mora et al. [1] extended ORCA for multi-robot navigation environments, accounting for non-holonomic constraints. Their work has been the basis of several multi-robot navigation approaches that account for the uncertainty present in real world environments [28], including approaches that focus on autonomous vehicles navigation [55]. In the same domain, some approaches have been proposed that not only address the collision avoidance problem, but also aim at generating robot trajectories that are smooth and comfortable. Authors in [51] propose an approach that combines model predictive control (MPC) with equilibrium point control to generate online trajectories for a robot in dynamic environments that not only avoids collisions but also prevents fast disturbances between consecutive MPC planning cycles. This approach reduces the variations in consecutive velocities as compared to ORCA. Other methods that use an MPC or MPC-inspired formulations on top of a collision avoidance method include [4] and our own previous work [16]. In general, such methods are computationally intensive, especially when it is necessary to model and predict multi-agent interactions in crowded environments, such as the ones addressed by *C-Nav*, while the quality of the solutions depends on the planning horizon.

Recent contributions have adopted a data-driven approach, taking advantage of the success of deep reinforcement learning methods on agent control tasks [65, 19, 7, 43, 44, 42]. These approaches allow agents to learn policies and navigate collision-free and autonomously in a variety of domains. However, these methods have difficulties dealing with highly dense scenarios and unseen situations [8, 41, 9], which are the types of environment that our *C-Nav* method targets.

Researchers have also studied multi-agent navigation in the context of human robot interaction, such as the works of Trautman et al. [66], Kretzschmar et al. [38] and Sisbot et al. [63], that address the problem of socially compliant mobile robot navigation.

All these works focus on achieving safe navigation, and some of them aim at incorporating smoothness and anticipation in the robot behaviors. These methods do not require explicit communication between the robots (although some of them require a training phase to work appropriately). However, what often is not explored is how efficient is the navigation of the agents, that is, how much time they take to reach their goals. As Figure 1(a) shows, pure collision avoidance may not be enough to move agents to their goals. Hence, a higher level of coordination between the agents might be needed to ensure not only

collision-free motion but also an efficient one. This is the problem that *C-Nav* addresses. In what follows, we review related work that focuses on achieving such coordination with and without explicit communication.

Coordinated Multi-Agent Navigation. Many approaches have been proposed to allow agents to coordinate their motions while moving to their goals. In some of these methods, there is no need for explicit communication between the agents. For example, when simulating the motion of pedestrians, coordination can be achieved through social norms [10], which can be embedded in the system [37] or can emerge through the interactions between the agents [67]. Other works such as [12] assume similar behaviors between the agents and use cognitive models of social interactions. Similarly, our work allows agents to compare motion features, without though relying on socio-psychological theory. Another approach for achieving coordinated navigation is by explicitly forming groups of agents that follow specific directions while maintaining cohesiveness among their members [34, 33]. A recent work addresses emergent group formation of different sizes based on proxemics [25].

Other works use the motions of the agents to communicate their intentions. For example, Mavrogiannis et al. [46] propose a framework for multi-agent navigation based on the concept of "social momentum", which is a way for an agent to take into account other agents' motions when planning its own movements. This approach allows each navigating agent to "read" the intentions of other agents based on their observed velocities. Using this information, the agent moves in a way that is "legible" to others while also progressing safely to its goal. This work has similarities with our proposed approach, such as the fact that agents consider both their goal progress as well as how they affect the other agents when making navigation decisions. A notable difference though, is that [46] assumes that motions of other agents are correlated with their intentions, which might not be the case in highly crowded environments where agents have to prioritize safety over performing legible actions. Our proposed approach, *C-Nav*, addresses navigation tasks in very constrained environments, where agents broadcast their intended motions to allow others to "read" them and take them into account when deciding what action to take.

In grid-like environments, some approaches encourage coordination by marking the cells in the grid with information which can be used by the agents to reduce congestion. In [32], each agent moves along neighbors with a similar goal, adjusting its path cost based on the agents' relative velocities. This idea is extended to account for congestion in [56]. Other approaches consider the generation of bounded suboptimal paths for multiple agents [5] and the kinematic constraints that emerge when dealing with embodied agents [30]. However, in most of these approaches, coordination is achieved by incorporating features in the environment that are assumed to be known by (or communicated to) all agents. Further, in all these approaches, the degree of coordination depends on the chosen resolution of the grid. In our approach, agents move in a continuous 2D environment. User-driven coordination has been studied in continuous 2D environment in [53], where agents can be guided to their goals to avoid conges-

tion. Our approach does not need external guidance and automatically generates coordinated goal paths for the agents.

Knepper and Rus [36] proposed a distributed cooperative collision avoidance method to find safe trajectories for robots in environments populated with humans and other robots. Their method is based on the concept of "civil inattention": in case of a potential collision, robots adapt their motion strategy depending on whether the other agent is engaged or not in the collision avoidance effort. In case of a potential collision between robots, it relies on explicit communication to coordinate the appropriate collision avoidance strategy. This adaptive strategy makes the robot motion more socially acceptable in human environments. Finally, coordination methods based on consensus between the agents have been proposed, for example, in [49, 48], which use graph-based methods and assume two-way communication between neighboring agents to reach consensus on formation control parameters, among others. Our method draws inspiration from these types of approaches, though it focuses more on large, distributed planning tasks. Most of these approaches consider only a limited number of agents, while *C-Nav* can scale up to hundreds of agents, as we show in Section 6.4.

3. Problem Formulation

In our problem setting, we assume there are n independent agents, $A_1, \ldots, A_n$, each with an individual start and goal position. We further assume that all agents use the same local collision avoidance routine. For simplicity, agents move on a 2D plane where there can be static obstacles O, approximated as line segments. We model each agent A_i as a disc with a fixed radius r_i. At timestep t, the agent A_i has a position $\mathbf{p}_i$ and moves with velocity $\mathbf{v}_i$ that is subject to a maximum speed, $v^{\max}$. Furthermore, A_i has a set of empirically defined preferred velocities $\mathbf{v}_i^{\text{pref}}$ (see Fig. 2) that indicate the agent's possible *intended* direction and speed of motion at a given timestep.

Definition 1. *Intended velocity,* $\mathbf{v}_i^{\text{intent}}$*: The intended velocity of agent* i *corresponds to the agent's current preferred velocity.*

A_i also has a goal velocity $\mathbf{v}_i^{\text{goal}}$ directed toward the agent's goal $\mathbf{g}_i$ with magnitude equal to $v^{\max}$. In the absence of any other agents and static obstacles, $\mathbf{v}_i^{\text{intent}} = \mathbf{v}_i^{\text{goal}}$, i.e., the intended velocity of A_i is equal to its goal velocity.

We assume that an agent can sense the radii, the positions, and the velocities of a subset of the agents, $\mathcal{N}$, composed by at most $|\mathcal{N}|$ agents within a limited fixed sensing range. We further assume that agents are capable of limited one-way communication. Specifically, each agent uses this capability to broadcast its unique ID and its intended velocity to its neighbors. This type of communication scales well, as it is not affected by the size of the agent's neighborhood.

Our task is to steer the agents to their goals without colliding with each other or with the environment, while reaching their goals as fast as possible. Since the agents navigate *independently* with only limited communication, this task has to be solved in a decentralized manner. Therefore, at each timestep t, we seek

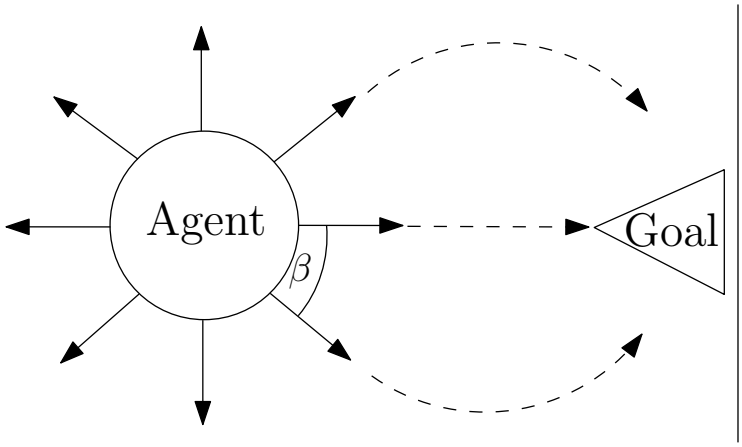


Figure 2: The set of actions in *C-Nav*: moving at 1.5 m/s with different angles with respect to the goal: $0°$, β, $-\beta$, $90°$, $-90°$, $180°$, $180° + \beta$ and $180° - \beta$. In our implementation, $\beta = 45°$. Dotted lines indicate the projection of the corresponding preferred velocities.

to find for each agent a new collision-free velocity that respects its geometric and kinematics constraints while progressing the agent towards its goal. To do so, we rely on the ORCA navigation framework [3]. At each timestep, ORCA takes as input an intended velocity $\mathbf{v}_i^{\text{intent}}$ and returns a new velocity $\mathbf{v}^{\text{new}}$ that is collision-free and as close as possible to $\mathbf{v}_i^{\text{intent}}$ by solving a low dimensional linear program. While ORCA guarantees a locally optimal behavior for each agent, it does not account for the aggregate behavior of all the agents. As ORCA agents have only a goal-oriented $\mathbf{v}_i^{\text{intent}}$, and do not account for each other's internal state (including their intended velocities) they may get stuck in local minima, unable to coordinate their motions, leading to large travel times and, subsequently, globally inefficient motions.

We postulate that if each agent is able to take into account the intended motions of its neighbors as well as its own, it could move in a way that reduces the travel time of its entire agent neighborhood. To do so, we propose *C-Nav*, a distributed coordination approach that allows each agent to choose an action from a set of preferred velocities at each timestep (Fig. 2), optimizing on a combination of its own goal progress as well as the progress of its neighbors. Throughout this paper, we use the terms action and preferred velocity interchangeably.

4. The *C-Nav* Approach

C-Nav is a *distributed coordination* framework for multi-agent navigation tasks. This type of coordination can be achieved when: 1) agents share their intended velocities via broadcast to their nearby agents and 2) agents use the information broadcasted by others, as well as their own goal progress, to make

better decisions on how to move and coordinate with each other without the need for explicit negotiation. With *C-Nav*, agents can select from a set of actions, adopted from our previous works [16, 15, 18] (see Fig. 2), in a way that helps their entire neighborhood of agents move to their goal locations. As we will show later in Section 6, this reduces the travel time of all the agents.

To achieve this, a *C-Nav* agent follows a three-step process. First, the agent simulates the execution of each of its actions for a number of timesteps into the future. For each simulated timestep, the agent evaluates the effect that a given action might have on its own goal progress, as well as on the motion of its most constrained neighbors, that is, neighbors whose observed velocities are most different from their broadcast intended velocities. Secondly, the agent selects an action that maximizes a reward function, $\mathcal{R}$, that evaluates its potential goal progress, as well as the progress of its most constrained neighbors. Finally, the preferred velocity $\mathbf{v}^{\text{pref}}$ corresponding to the chosen action is passed to the collision-avoidance framework, ORCA, which computes a collision-free velocity $\mathbf{v}^{\text{new}}$ to be used during the next timestep.

Algorithm 1 outlines *C-Nav*. For each agent that has not reached its goal, a new action is computed every few timesteps, on average every 0.2 seconds (line 4), empirically determined. In each new update, the agent computes which neighbors are most constrained to move (line 5), and uses this information to evaluate all of its actions (line 7). After this evaluation, the best action is selected (line 9) as measured by the above described reward function $\mathcal{R}$. Finally, its intended velocity $\mathbf{v}^{\text{intent}}$ is broadcast to the agent's neighbors (line 10) and is also mapped to a collision-free velocity $\mathbf{v}^{\text{new}}$ via the ORCA framework (line 12), which is used to update the agent's position (line 13). The cycle repeats until the agent reaches its goal.

Algorithm 1: The C-Nav(i) framework for agent i

```
1:  Input: agent i
2:  start the navigation
3:  while not at the goal do
4:    if UpdateAction(t) then
5:      C_rank ← GetMostConstrainedNeighs(i)
6:      for all a ∈ Actions do
7:        R_a ← SimMotion(i, a, C_rank)
8:      end for
9:      v^pref ← arg max_{a∈Actions} R_a
10:     broadcast ID and v^intent to nearby agents
11:   end if
12:   v^new ← CollisionAvoidance(v^intent)
13:   p^t ← p^{t+1} + v^new · Δt
14: end while
```

In what follows, we describe how *C-Nav* agents choose their most constrained neighbors, and how are these neighbors taken into account when evaluating

the available actions. Finally, we show how *C-Nav's* reward function balances between two objectives: reducing motion constraints in the neighbors and increasing the goal progress for the agent.

4.1. Determining constrained neighbors

With information obtained by sensing (radii, positions and velocities) and via one-way communication (IDs and intended velocities) from all the neighbors within the sensing range, each agent estimates which nearby agents are the most constrained ones. Specifically, agents use the intended velocities of their neighbors to evaluate how constrained their motion is and, thus, determine neighbors that are more likely to slow down the overall progress of the crowd. By reducing the constraints of these neighbors, i.e. by being *polite* towards them, the time- and energy- efficiency of the system increases.

Definition 2. *Polite action: For a given neighbor j, the politeness of an agent's action a is measured by how much the neighbor's $\mathbf{v}_j^{\text{intent}}$ would be impeded during the next timestep, if the agent took action a. Formally, given that the maximum speed that the neighbor j can attain is $v^{\max}$, the politeness $\mathcal{P}_{a,j}$ is given as follows:*

$$\mathcal{P}_{a,j} = v^{\max} - \| \mathbf{v}_j^{\text{intent}} - \mathbf{v}_j^{\text{new}} \|. \tag{1}$$

It is important to distinguish our definition of *politeness*, in the context of *C-Nav*, from the notion of reciprocity of the underlying multi-agent navigation framework, ORCA. ORCA's reciprocity focuses on the low-level planning of the agents and its sole purpose is to prevent agents from colliding with each other, while the *politeness* defined above focuses on a higher-level planning of the agents and it is intended to avoid or reduce congestion. Hence, both behaviors are complementary.

Algorithm 2 details the constraint evaluation procedure, GetMostConstrainedNeighs(i), for agent i. First, each agent compares a neighbor's intended motion with its observed velocity (line 7). The larger the difference, the more likely it is that the neighbor's motion is impeded. To avoid circular dependencies which can give rise to deadlocks, each agent only considers neighbors that are closer than itself to its goal (line 5). This ensures that no two agents with the same goal will simultaneously defer to each other. The agent keeps two lists, C and D, which keep track of the neighbor ID and quantify the constraints of each neighbor, respectively. After all neighbors have been evaluated, C is sorted in descending order based on the values in D, and a list C_{rank} of the indices of the sorted neighbors is returned (line 10-12).

Once the agent computes a ranking of the most constrained neighbors, it can use this information to bias the action selection towards velocities that, on one hand, move the agent closer to its goal while, on the other, help nearby agents to move according to their intended motions.

4.2. Improving neighborhood motion

Agents can choose a preferred velocity from the set of actions shown in Figure 2, which allows agents to choose velocities that are uniformly distributed

Algorithm 2: GetMostConstrainedNeighs(i): Compute most constrained neighbors for a given agent i

1: **Input:** agent i (includes list of neighbors $\mathcal{N}(i)$, goal position $\mathbf{g}_i$ and current position $\mathbf{p}_i$)
2: **Output:** C_{rank}, list of indices of the most constrained neighbors
3: $C \leftarrow [\,]$, $D \leftarrow [\,]$
4: **for all** $j \in \mathcal{N}(i)$ **do**
5: **if** $\|\mathbf{g}_i - \mathbf{p}_j\| < \|\mathbf{g}_i - \mathbf{p}_i\|$ **then**
6: $C.insert(j)$
7: $D.insert(\| \mathbf{v}_j^{\text{intent}} - \mathbf{v}_j^{\text{new}} \|)$
8: **end if**
9: **end for**
10: *Sort list C by value D in descending order*
11: $C_{rank} \leftarrow C$
12: return C_{rank}

in the space of directions. To evaluate each available action, an agent simulates its execution for a number of timesteps and evaluates two metrics: its potential progress towards its goal and its effect in the motion of its k most constrained neighbors ($0 \leq k \leq |\mathcal{N}|$). This procedure is called SimMotion*(i, a, C_{rank})* in Algorithm 1 (line 7). In practice, SimMotion executes each action for two time steps (less than a quarter of a second). We empirically found this small time window to be sufficient, as our goal here is to assess how the action affects the k most constrained neighbors of the agent rather than trying to capture complex type of dynamics. Algorithm 3 outlines the overall procedure.

Motion simulation

As a first step, for each given action, an agent uses the ORCA framework to simulate the changes in its neighborhood (line 4), updating the velocities and positions of itself and its neighbors for each timestep within a fixed time horizon T (line 3). Note that in very crowded areas (such as in Figure 3(a)), agents often have no control over their own motions, as they are being pushed by other agents to avoid collisions. Hence, simulating the dynamics of all the agent's neighbors often results in the same velocity for all simulated actions (Figure 3(b)). This prevents the agent from selecting a velocity that improves the motion of its most constrained neighbors. Because of this, in *C-Nav* the agent considers in its simulation only the neighbors that are closer to its goal than itself (Figure 3(c)), 'ignoring' the agents that are behind it with respect to its goal (Figure 3(d)). Even if the best valued action is not currently allowed, we expect that the neighboring agents will eventually try to relax the constraints that they impose on the agent, and enable the agent to make progress towards its goal.

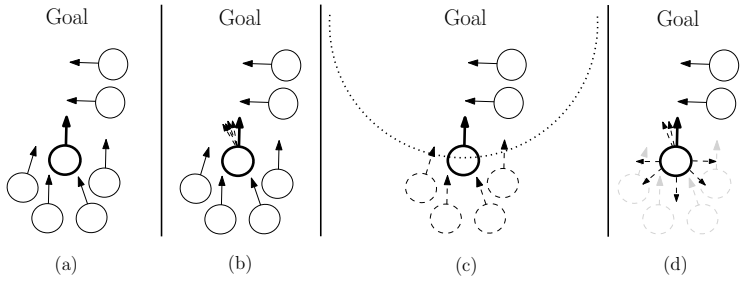


Figure 3: Motion simulation of *C-Nav* for the bold agent in crowded areas. (a) Initial crowded conditions. (b) If the bold agent takes into account all of its neighbors in the motion simulation step (line 3 of Algorithm 3), all of its simulated actions would result in similar collision-free velocities (to the left of its goal-oriented path). (c) With *C-Nav*, the bold agent only considers the neighbors that are closer to its goal. (d) Because the bold agent "ignores" agents coming from behind, it assumes that it can move backwards unconstrained (for example, to avoid introducing motion constraints to the two agents moving to the left). Even if such movement is not immediately allowed by the collision-avoidance mechanism, it might eventually be feasible if the "ignored" agents behave politely towards the bold agent.

Algorithm 3: SimMotion*(i, a, C_{rank})*

1: **Input:** agent i (includes list of neighbors $\mathcal{N}(i)$), integer $a \in Actions$, sorted list of indices C_{rank}
2: **Output:** R_a, estimated value of action a
3: **for** $t = 0, \ldots, T-1$ **do**
4: simulate evolution of neighborhood dynamics
5: **if** $t > 0$ **then**
6: **for all** $j \in \mathcal{N}(i)$ **do**
7: **if** $\text{rank}(j \in C_{rank}) < k$ **then**
8: $\mathcal{R}_a^c \leftarrow \mathcal{R}_a^c + v^{\max} - \| \mathbf{v}_j^{\text{intent}} - \mathbf{v}_j^{\text{new}} \|$
9: **end if**
10: **end for**
11: **end if**
12: $\mathcal{R}_a^g \leftarrow \mathcal{R}_a^g + \mathbf{v}_i^{\text{new}} \cdot \frac{\mathbf{g}_i - \mathbf{p}_i}{\|\mathbf{g}_i - \mathbf{p}_i\|}$
13: **end for**
14: $\mathcal{R}_a^g \leftarrow \frac{\mathcal{R}_a^g}{T \cdot v^{\max}}, \mathcal{R}_a^c \leftarrow \frac{\mathcal{R}_a^c}{(T-1) \cdot k \cdot v^{\max}}$
15: $\mathcal{R}_a \leftarrow (1-\gamma) \cdot \mathcal{R}_a^g + \gamma \cdot \mathcal{R}_a^c$
16: return $\mathcal{R}_a^g$

Neighborhood influence

After simulating a specific action for the given time horizon, the agent can estimate how this action affects each of its k most constrained neighbors.

For a given neighbor j, it computes this based on the difference between j's predicted collision-free velocity $\mathbf{v}_j^{\text{new}}$ and its communicated intended velocity $\mathbf{v}_j^{\text{intent}}$ (line 8).

Motion evaluation

To decide what motion to perform, the agent aims at minimizing the amount of constraints imposed to its neighbors, while also ensuring progress towards its own goal. Our reward function balances these two objectives, by taking a linear combination of a *goal-oriented* and a *constrained-reduction* component (Eq. 2). Each component has an upper bound of 1 and a lower bound of -1 and is weighted by the *coordination-factor* γ.

$$\mathcal{R}_a = (1-\gamma) \cdot \mathcal{R}_a^g + \gamma \cdot \mathcal{R}_a^c \tag{2}$$

The *goal-oriented* component $\mathcal{R}_a^g$ computes, for each timestep in the time horizon, the scalar product of the collision-free velocity $\mathbf{v}_i^{\text{new}}$ of the agent with the normalized vector which points from the position $\mathbf{p}$ of the agent i to its goal $\mathbf{g}_i$. This component encourages preferred velocities that lead the agent as quickly as possible to its goal. Formally:

$$\mathcal{R}_a^g = \frac{\sum_{t=0}^{T-1} \left(\mathbf{v}_i^{\text{new}} \cdot \frac{\mathbf{g}_i - \mathbf{p}_i}{\|\mathbf{g}_i - \mathbf{p}_i\|} \right)}{T \cdot v^{\max}} \tag{3}$$

The *constrained-reduction* component $\mathcal{R}_a^c$ averages the amount of constraints introduced in the agent's k most constrained neighbors. This component promotes preferred velocities that do not impede these k neighbors. More formally:

$$\mathcal{R}_a^c = \frac{\sum_{t=1}^{T-1} \sum_{j \in C_{rank}} \mathcal{P}_{a,j}}{(T-1) \cdot k \cdot v^{\max}} \tag{4}$$

An agent which only aims at maximizing $\mathcal{R}_a^g$ would be selfish and it would not consider the effect that its actions have on its neighbors. On the other hand, if the agent only tries to maximize $\mathcal{R}_a^c$, it might have no incentive to move towards its goal, which means it might never reach it. Therefore, by maximizing a combination of both components, the agent coordinates its goal-oriented motion with that of its neighbors, resulting in lower travel times for all agents. Overall, the low communication and computation overhead that *C-Nav* imposes over the ORCA underlying navigation framework allows our approach to simulate hundreds of agents in real time, as well as its application to multi-robot navigation tasks.

To better highlight the advantages of *C-Nav*, Figure 4 compares the behavior of vanilla ORCA and *C-Nav* in a small scenario with two agents, whose goal is to reach the other side of a hallway. The agents' initial positions is shown in Figure 4 (a), with agent 1 initially closer to the goal than agent 2. With ORCA, agent 2, coming full speed from behind, introduces constraints into the goal

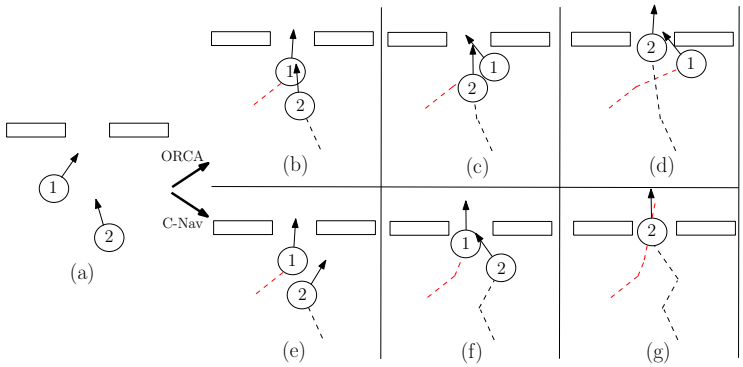


Figure 4: Example executions of ORCA (top) and *C-Nav* (bottom), in a scenario with two agents with identical goal positions. (a) Agents in their starting positions. (b) agents using ORCA have a single preferred velocity towards their goals; (c) agent 1 is forced to move to the right to avoid collision with agent 2 coming from behind, even though agent 1 is closer to the goal; (d) agent 2 overpasses agent 1, reaching its goal first; (e) agent 2, using *C-Nav*, selects a preferred velocity that allows agent 1 to move unconstrained to its goal position; (f) agent 2 resumes its goal oriented motion only when it does not introduce constraints in the motion of agent 1; (g) agents 1 and 2 reach their goal faster than with ORCA.

oriented motion of agent 1, forcing the latter to move aside to avoid colliding with the former (Figures 4 (b) and (c)). Only after agent 2 has overtaken agent 1 can the latter resume its goal-oriented motion (Figure 4 (d)). This behavior is not only inefficient in terms of travel time, but it also appears "impolite" from agent 2 towards agent 1. In contrast, with *C-Nav*, the communication of intended velocity as well as the proposed reward function allows agent 2 to evaluate and choose a preferred velocity that does not introduce constraints in the motion of agent 1, enabling the latter to freely move to its goal (Figures 4 (e) and (f)). Finally, this polite behavior not only respects their original order in terms of goal distance, but also allows the agents to reach their goals faster than with ORCA (Figure 4 (g)).

5. Theoretical Analysis

We focus our theoretical analysis on showing the conditions under which *C-Nav* agents can avoid livelocks. A livelock corresponds to executing a series of repeated motions that do not move the agent to its goal. In *C-Nav*, a livelock would occur if two or more agents repeatedly switch from goal-oriented motions to polite motions that move the agents away from their goals, which would prevent the progress of the agents. We show that there is some value of γ (Eq. 2), where the probability of livelocks to occur equals zero in scenarios where all agents share the same goal (e.g., the CONGESTED scenario in Figure 5)

For the purpose of this analysis, we assume that after an agent reaches the goal, it is removed from the environment. Let A_α correspond to the agent that, at any given time, is closest to the goal.

Lemma 1. *At any time, A_α is able to choose an action that maximizes its progress to the goal, without deferring to the motion of other agents.*

Proof: The proof follows from the fact that an agent only accounts for neighbors that are closer than itself to the goal for the evaluation of constraints and for motion simulation purposes (Section 4.2). As A_α ignores agents coming from behind, and there are no agents closer than A_α to the goal, then for all of its actions a, $\mathcal{R}_a^c = 0$, which means that A_α will optimize only the value of the action's goal progress $\mathcal{R}_a^g$. To ensure that A_α has an incentive to reach the goal, $\mathcal{R}_a^g$ must be greater than zero for at least one of the actions. Hence, as long as $\gamma < 1$, A_α will choose the action with the collision-free velocity that maximizes its progress to the goal. ■

Lemma 2. *Any agent A_i, where $A_i \neq A_\alpha$ and $A_\alpha \in \mathcal{N}(A_i)$, can choose an action a' that moves it backwards from its goal without introducing constraints into A_α.*

Proof: To allow A_α to maximize its progress to the goal, A_i should always choose an action a' that does not introduce constraints into A_α ($a' = \arg\max_{a \in Actions} \mathcal{R}_a^c$). Such an action a' always exists; in the worst case, a' corresponds to the preferred velocity backwards from the goal. As A_i ignores agents coming from behind (Section 4.2), it assumes it can freely move in this direction. This backwards velocity moves A_i away from A_α, minimizing the difference between A_α's intended velocity and its collision-free velocity. For A_i to choose this action, it must hold that $a' = \arg\max_{a \in Actions} \mathcal{R}$. Although there is no single value of γ that guarantees this condition for all possible values of $\mathcal{R}_a^c$ and $\mathcal{R}_a^g$, A_i will eventually choose action a' as $\gamma \to 1$ (see Eq.2). As each agent A_i chooses a', it will not introduce constraints into A_α's motion, which will allow A_α to move as if it was the only agent in the system and to reach the goal. Once this occurs, another agent takes the role of A_α, until all agents reach the goal. ■

From Lemmas 1 and 2, the following Theorem holds:

Theorem 1. *In environments where agents share a common goal, the probability of livelocks in* C-Nav *reaches 0 as γ asymptotically approaches 1. Under these conditions, all agents are guaranteed to reach their goals.*

6. C-Nav evaluation in simulation

6.1. Experimental Setup

We implemented *C-Nav* in C++. Results were gathered on an Intel Core i7 at 3.5 GHz. Each experimental result is the average over 100 simulation runs. In all our runs, we updated the positions of the agents every $\Delta t = 50\,\mathrm{ms}$ and

set the maximum speed $v^{\max}$ of each agent to 1.5 m/s and its radius to 0.5 m. Agents could sense other agents within a 15 m radius, and obstacles within 1 m. To avoid synchronization artifacts, agents are given a small random delay in how frequently they can update their $\mathbf{v}^{\text{pref}}$ (with new $\mathbf{v}^{\text{pref}}$ decisions computed every 0.2 s on average). This delay also gives ORCA and *C-Nav* a few timesteps to incorporate sudden velocity changes before the actions are evaluated. Small random perturbations were added to the preferred velocities of the agents to prevent symmetry problems [3]. To simulate the communication of the intended velocity, we allowed agents to access the preferred velocity of their nearby agents.

6.2. Performance Metric

To evaluate the performance of *C-Nav*, we use the interaction overhead metric proposed by Godoy et al. [15], which measures the time that the agents take to reach their goals compared to their travel time if they could move unconstrained from their initial to their goals positions (the upper bound of their theoretical minimum travel time). This metric is independent of the distances between the agents' initial and goal positions (which depend on the specific scenario), and it allows us to compare the time performance of the evaluated methods. It answers the question of how much time the agents spent reacting to other agents rather than moving to their goals. It also allows us to compare the complexity of the different scenarios, as high values of interaction overhead are indication of scenarios with more complex multi-agent interactions, compared to scenarios with low values. For completeness, we include the formal definition of interaction overhead below.

Definition: Interaction Overhead. The interaction overhead is the difference between the travel time of the set of agents A, as measured by Eq. 6, and their hypothetical travel time if all the agents could follow their shortest paths to their goals at maximum speed without interacting with each other, i.e.:

$$Interaction\ Overhead = TTime(A) - MinTTime(A) \tag{5}$$

where $TTime(A)$ accounts for the global travel time of all agents in A. To evaluate this travel time, we could consider the travel time of the last agent that reaches its goal in each scenario. However, this value would not provide us with any information regarding the travel time of all the other agents. Instead, $TTime(A)$ accounts for the average travel time of all the agents in A and its spread. Formally:

$$TTime(A) = \mu\left(TimeToGoal(A)\right) + 3\ \sigma\left(TimeToGoal(A)\right) \tag{6}$$

where $TimeToGoal(A)$ is the set of travel times of all agents in A from their start positions to their goals, and $\mu(\cdot)$ and $\sigma(\cdot)$ are the average and the standard deviation (using the unbiased estimator) of $TimeToGoal(A)$, respectively. If the travel times of the agents follow a normal distribution, then $TTime(A)$ represents the upper bound of $TimeToGoal(A)$ for approximately 99.7% of the agents. Even if the distribution is not normal, at least 89% of the times will fall within three standard deviations (Chebyshev's inequality).

Following the same reasoning, $MinTTime(A)$ accounts for the average and spread of the theoretical minimum travel time of the set of agents A, evaluated as follows:

$$MinTTime(A) = \mu\left(MinimumGoalTime(A)\right) + 3\sigma\left(MinimumGoalTime(A)\right) \quad (7)$$

where $MinimumGoalTime(A)$ is the set of travel times for all agents in A, if they could follow their shortest route to their goals, unconstrained, at maximum speed.

The interaction overhead metric allows us to evaluate the performance of *C-Nav* from a theoretical standpoint in each of the navigation scenarios. An interaction overhead of zero represents a lower bound on the optimal travel time for the agents, and it is the best result that any optimal centralized approach could potentially achieve.

6.3. Simulation scenarios

To evaluate *C-Nav* we used a variety of scenarios, with different numbers of agents and, in some cases, static obstacles. Figure 5 shows the different simulation scenarios. These include:

(a) INTERSECTION: 80 agents in four perpendicular streams meet in an intersection (Fig 5(a)). This scenario, besides its complexity, reflects crowd congestion that could potentially occur in real life such as at shopping centers or busy pedestrian crossings;
(b) CROWD: 300 randomly placed agents must reach their randomly assigned goal positions, while moving inside a squared room (Fig 5(b));
(c) CONGESTED: 32 agents are randomly placed close to the narrow exit of an open hallway and must escape the hallway through this exit (Fig. 5(c));
(d) LINE: 4 agents placed in a line next to a narrow exit must reach the other side of this exit (Fig 5(d));
(e) CIRCLE: 128 agents walk to their antipodal points on a circle (Fig 5(e));
(f) BIDIRECTIONAL: two groups of 9 agents each move in opposite directions inside a corridor (Fig. 5(f)).

6.4. Results

We first study how *C-Nav* compares against two popular multi-agent navigation frameworks, ORCA [3] and Social Forces [27]. Next, we evaluate how *C-Nav*, as a communication-based distributed coordination approach, compares against a learning-based method for action coordination (ALAN [15]). We then evaluate the contribution of the constraint-reduction (politeness) component of *C-Nav* to its overall performance. We also evaluate *C-Nav* in a warehouse-like environment, one of our target domain problems, and lastly we compare the energy efficiency of *C-Nav* to ORCA in the different simulation scenarios.

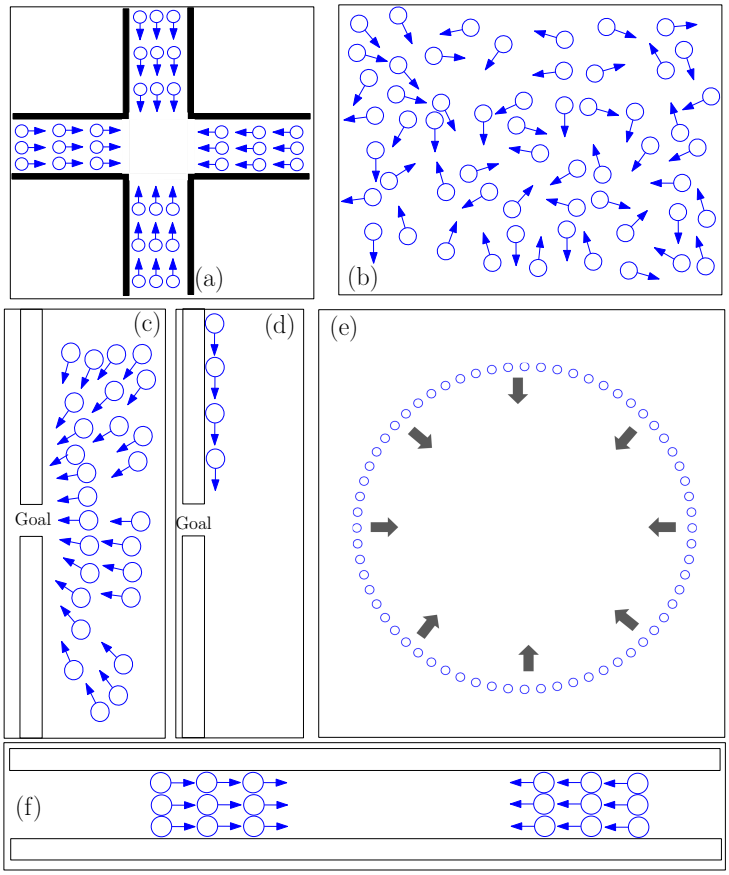


Figure 5: Simulated scenarios:(a) Intersection, (b) Crowd, (c) Congested, (d) Line, (e) Circle and (f) Bidirectional.

6.4.1. Comparison of C-Nav to other navigation approaches

We evaluated the interaction overhead times in all scenarios depicted in Figure 5. Results can be seen in Figure 6. The interaction overhead of *C-Nav* is significantly lower than ORCA's in all cases, which indicates that by considering information about their neighborhood, agents can improve their time-efficiency. Even in scenarios where agents are constrained by other agents and static obstacles (such as in the Bidirectional scenario), *C-Nav* is able to significantly improve the time-efficiency of the agents. In terms of qualitative results, we observe an emergent behavior in the Bidirectional and Circle scenarios, where agents going in the same direction form lanes. Such lanes reduce the constraints in other agents leading to more efficient simulations. Note that, in the Intersection scenario, *C-Nav* agents take only about one third of the time that ORCA agents need in order to reach their goals.

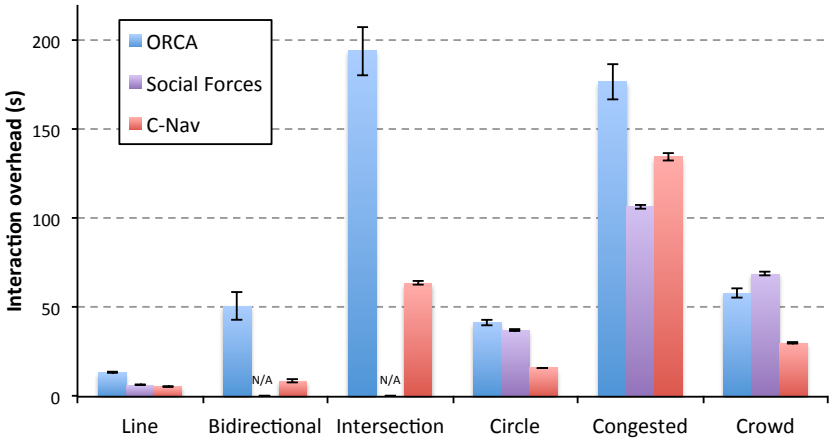


Figure 6: Performance comparison between ORCA, Social Forces and *C-Nav*. In all but the CONGESTED scenario, agents using our coordination approach have the lowest overhead times. The error bars correspond to the standard deviation.

In most scenarios, *C-Nav* outperforms the Social Forces approach in terms of interaction overhead times, although less noticeably (but still significantly) in the LINE scenario. The only exception occurs in the CONGESTED scenario, where the repulsion force among agents [27] creates enough space between them to allow individual agents to quickly exit the hallway, reaching their goals faster than the other two methods.

In the LINE, CIRCLE and CONGESTED scenarios, the Social Forces approach outperforms ORCA in interaction overhead time. On the other hand, the Social Forces approach is unable to move agents to their goals in the BIDIRECTIONAL and INTERSECTION scenarios. Here, agents get stuck in a deadlock due to their conflicting goals. ORCA, however, is able to move the agents to their goals even in these very constrained environments. Further, the Social Forces approaches does not provide the collision-free guarantees of ORCA. As multi-robot navigation is one of the main domain applications of *C-Nav*, it is critical to guarantee safe navigation among the robots. Due to all these reasons, we use ORCA as *C-Nav's* underlying navigation framework.

6.4.2. Results of coordination strategy

In previous work, we have studied different coordination methods to increase the time efficiency of agents in navigation tasks. In one of such methods, called ALAN [15], agents learn to coordinate their motions via action sampling, evaluating their individual goal progress without communication. *C-Nav*, on the other hand, can be seen as a more "social" method: agents coordinate their motions via communication of intended velocities, and evaluating the impact of their actions on their nearby agents. We compare these two approaches for coordinating the motions of the agents: *C-Nav*, as a communication-based

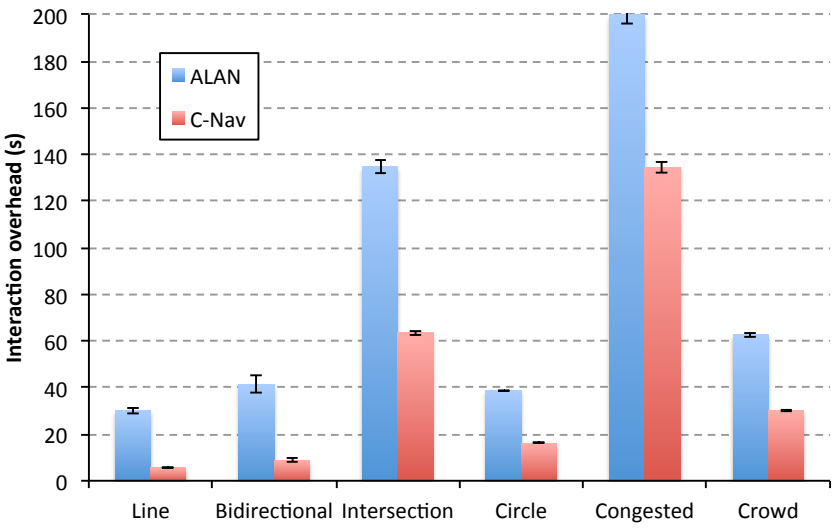


Figure 7: Performance comparison between *C-Nav* and a learning-based coordinated action selection method (ALAN). In all scenarios, agents using our distributed coordination approach have the lowest overhead times. The error bars correspond to the standard deviation.

distributed coordination approach, and ALAN as a learning-based method for coordinated action selection. Results, which can be observed in Figure 7, show that *C-Nav* achieves lower interaction overhead than ALAN in all scenarios. *C-Nav* agents redefine the concept of 'polite' behavior, as compared to ALAN [15], to explicitly consider their nearby agents' intended motions. With this extra piece of information, each agent determines the optimal action that maximizes both its own goal progress as well as its neighbors'. Hence, *C-Nav* agents take actions that benefit their entire neighborhood. This 'polite' behavior minimizes the occurrence of congestion which, in long term, minimizes the travel time of all agents.

So far, we have shown that *C-Nav* outperforms other navigation approaches, as well as an action selection method based on the same underlying collision avoidance framework. An interesting evaluation is to determine how much of *C-Nav*'s performance improvement is due to its 'polite' behavior component, and how much of it is due to the reciprocal behavior component of ORCA. To evaluate this, we removed the reciprocity feature of ORCA (each agent now exerts full effort to avoid collisions, instead of only half of this effort as in vanilla ORCA) and compared the interaction overhead of this 'No-Reciprocity ORCA' (NR-ORCA) with and without *C-Nav* running on top. Table 1 summarizes the interaction overhead times for both approaches, which shows that isolating the politeness component from the reciprocity of ORCA still results in a significant performance improvement of *C-Nav* over this version of ORCA. Differences in

Scenario	NR-ORCA	NR-ORCA with C-Nav
Line	11.9±0.7	5±0.2
Bidirectional	19.1±2.1	7.7±0.8
Intersection	358.6±20.4	78.3±1.1
Circle	31.7±0.6	17.5±0.2
Congested	230.8±4.7	192.1±2.3
Crowd	45.5±0.8	36.2±0.4

Table 1: Interaction overhead (in seconds) for both NR-ORCA (ORCA with reciprocity removed) and NR-ORCA with *C-Nav* in all scenarios in Figure 5 over 100 trials.

each case are statistically significant (t-test with $p < 0.001$).

6.4.3. Results of constraint-reduction component

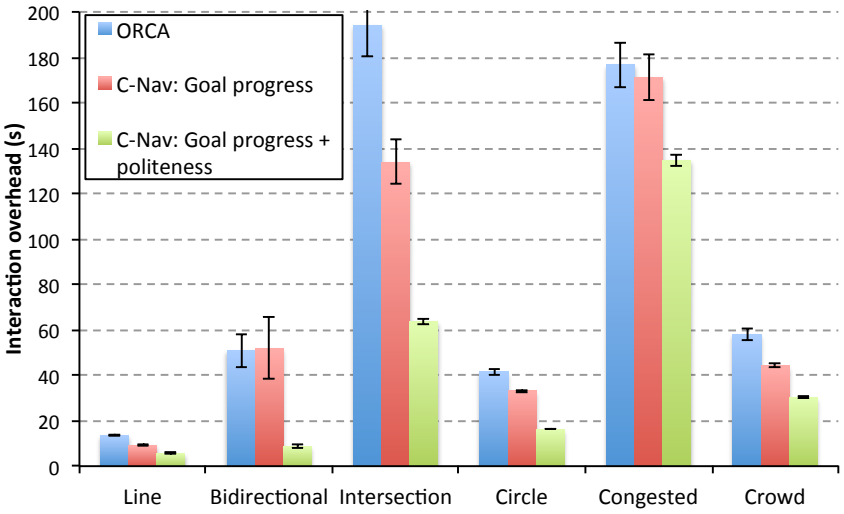


Figure 8: Performance comparison between ORCA (in blue), "selfish" *C-Nav* using only the goal progress as reward for the actions (in red), and "polite" *C-Nav* using both the goal progress and constraint-reduction components (in green). The error bars correspond to the standard deviation.

To evaluate the contribution of *C-Nav*'s *constraint-reduction* component, $\mathcal{R}_a^c$, to *C-Nav's* performance, we compared the interaction overhead obtained with just the *goal-progress* component, $\mathcal{R}_a^g$ (Eq. 3), which we call "selfish" *C-Nav*, to the one obtained using both the *goal-progress* and the *constraint-reduction* (politeness) components of the full reward function shown in Eq.2, which we call "polite" *C-Nav's*. Agents in both versions of *C-Nav's* have the option to

choose preferred velocities other than the goal-oriented ones which can lead to more promising collision-free velocities, as measured by the corresponding reward functions. We compared these two versions of *C-Nav's* against ORCA where, unlike "selfish" *C-Nav*, the agents' $\mathbf{v}^{\text{intent}}$ is always equal to their $\mathbf{v}^{\text{goal}}$. Results for all scenarios can be seen in Figure 8.

In all scenarios, "polite" *C-Nav* agents that try to reduce the motion constraints of their neighbors were able to reach their goals faster, on average, than "selfish" agents that only considered their goal progress when making action decisions. This difference is most noticeable in the INTERSECTION, BIDIRECTIONAL and CIRCLE scenarios, where the distributed coordination achieved by *C-Nav*'s constraint-reduction component (see Eq. 4) translates into a significant reduction of congestion and the consequent reduction of interaction overhead.

It is worth noting that even with "selfish" agents (using only the goal-oriented component), *C-Nav* outperformed ORCA in four out of six scenarios, and exhibited the same performance with ORCA in the BIDIRECTIONAL and CONGESTED scenarios. In these two cases, as agents have to move through obstacle-constrained areas for at least part of the navigation task (the corridor in the BIDIRECTIONAL and the narrow exit in the CONGESTED), the extra actions of *C-Nav* are unable to find alternative goal paths that would reduce their travel time. On the other hand, in scenarios such as the CIRCLE and the CROWD where agents have more space to maneuver, even pure selfish behavior is enough to move the agents to their goals faster than with ORCA.

6.4.4. Results in warehouse-like environment

One of the target domains of *C-Nav* is automated warehouses, where multiple robots have to navigate to dedicated goals in order to accomplish their tasks. To assess the potential advantages of our approach in such a domain, we evaluated *C-Nav* in the warehouse-like environment shown in Figure 9 and compared its performance to ORCA. The environment was obtained from [39] (used for benchmarking multi-agent path finding methods).

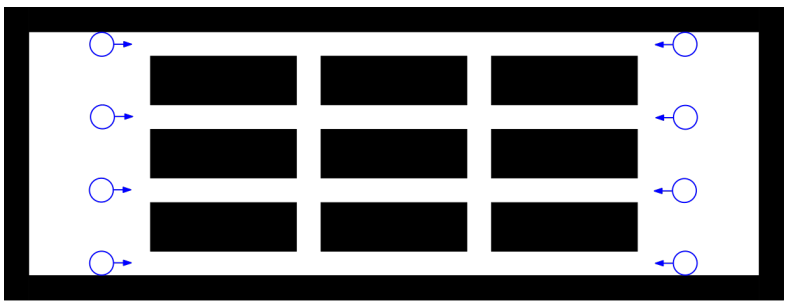


Figure 9: Warehouse-like scenario, of dimensions 10 × 30 meters, from [39], indicating the initial positions of the agents.

We initially placed 4 agents in the left and the right open areas in the environment (in total, 8 agents), and set their goals in the opposite positions in

the environment (which also corresponds to the initial position of another agent). To reach its goal, each agent must interact with another agent approaching from the opposite direction, in corridors that allow only one agent to pass at a time. It is worth noting that the agents lack any global roadmap and only rely on their local navigation framework to reach to their goals. As a result, they can easily get stuck behind static obstacles. We run 100 iterations of ORCA and *C-Nav*, and the resulting interaction overhead times are 1169.2 s $\pm$23.3 *s* for ORCA and 368.4 s $\pm$36.4 *s* for *C-Nav*. This noticeable difference shows that the politeness component of our approach has a significant effect in the agents' interactions. Further, in 18 of the 100 trials run, ORCA agents were not able to reach their goals, while *C-Nav* was able to successfully move the agents to their destinations in all of the trials run. This highlights the complexity of the environment and the benefits of using our proposed approach to solve complex situations.

Using the same environment, we also evaluated how *C-Nav* performed when agents had more than one goal. Specifically, we gave each agent one subgoal: the opposite position in the environment as before, and one final goal: its initial position. Therefore, each agent had to navigate back and forth in the warehouse through the corridors. Each experiment was run for 100 trials. In this setup, ORCA agents arrived to their goal positions in only 8 of the 100 trials run, having an interaction overhead of 1731.4 s $\pm$29.5 *s*, as compared to 100% goal reachability with *C-Nav* and 284.7 s $\pm$23.4 *s* of interaction overhead. This shows that the advantages of using *C-Nav* also translate to real-world domains.

6.4.5. Results in energy efficiency

Energy efficiency is critical in robot applications. To assess how *C-Nav* performed in this context, we used the metric proposed in [16] that measures the amount of energy consumed by the agents while navigating. Specifically, we approximate the power expected to be consumed by an agent i, while it is moving, as follows:

$$Energy(i) = b + c \cdot \|\mathbf{v}_i^{\text{new}}\|^2. \tag{8}$$

In Equation 8, b corresponds to the power consumed by the agent due its processing and sensing capabilities, while $c \cdot \|\mathbf{v}_i^{\text{new}}\|^2$ corresponds to the kinetic energy spent by the agent while moving, proportional to the agent's speed squared. Given that $\sqrt{b/c}$ denotes the optimal agent speed in terms of energy per second expended, in our implementation we set $b = 2.25$ and $c = 1$ resulting in an optimal speed equal to v^{max}. The results in Table 2 show the total instantaneous energy, averaged over all agents for both *C-Nav* and ORCA, over 100 trials. In all but the CONGESTED scenario, *C-Nav* agents expend less energy than agents using ORCA (t-test with $p < 0.001$). In the CONGESTED scenario, *C-Nav* agents backtrack often to alleviate congestion developing near the narrow exit, spending more energy on backwards motions than ORCA agents, which do not show this behavior. This result highlights another advantage of using *C-Nav* in multi-robot navigation tasks.

Scenario	ORCA	*C-Nav*
Line	553.9	378.3
Bidirectional	2761.2	2170.7
Intersection	6602.1	3498.9
Circle	16365.6	15832.4
Congested	6770.6	7091.3
Crowd	1920.2	1706.9

Table 2: A comparison of the average energy (in $J \cdot Kg^{-1} \cdot s^{-1}$) expended by the *C-Nav* agents in all scenarios in Figure 5. Smaller numbers denote less energy expended by the agents.

Scenario	ORCA	*C-Nav*
Line	2.5	3.0
Bidirectional	5.2	6.1
Intersection	2.2	3.3
Circle	6	6.4
Congested	1.3	1.2
Crowd	2.8	3.2

Table 3: A comparison of the energy efficiency (in $J \cdot meters \cdot Kg^{-1} \cdot s^{-2}$) of ORCA and *C-Nav* agents in all scenarios in Figure 5. Larger numbers denote more energy efficient agents.

While Table 2 focuses on the total instantaneous energy expenditure of the agents, it does not communicate the amount of energy expended by an agent while attempting to move toward its goal. To address this, we borrow ideas from our previous work in [16] and measure the ratio between the progress of the agent towards its goal, and the amount of energy expended, per second:

$$\textit{EnergyEff(i)} = \frac{Progress(i)}{Energy(i)}, \tag{9}$$

where *Progress(i)* is defined as:

$$Progress(i) = \mathbf{v}_i^{\text{new}} \cdot \frac{\mathbf{g}_i - \mathbf{p}_i}{\|\mathbf{g}_i - \mathbf{p}_i\|}. \tag{10}$$

Therefore, Eq. 9 measures the ratio between the *progress* of the agent towards its goal and the *energy* that the agent spent in such motion. We summed the *EnergyEff(i)* of each agent during its lifespan, and then computed the average energy efficiency across all agents over 100 trials. The results, shown in Table 3, indicate that, again, in all but the CONGESTED scenario, *C-Nav* agents are more efficient than ORCA agents, moving closer to their goals per energy expended at each second (t-test with $p < 0.001$).

7. Analysis

In this section, we analyze various aspects of the proposed *C-Nav* approach. We begin by analyzing the role of *C-Nav* in minimizing the total travel time of the agents. Then, we focus on its runtime complexity as well as on how it scales with the number of agents present in the environment. We also perform sensitivity analysis of *C-Nav* with respect to the value of the coordination factor (γ) used. Finally, we evaluate the sensibility of the results with respect to the number k of constrained neighbors considered by each agent during action evaluation.

7.1. Runtime Complexity

The runtime performance of *C-Nav* is dominated by its action-selection routine (Alg. 1, lines 5-9), as it is necessary to simulate the motion of the agent and its neighbors for each possible action, for the small time horizon considered ($T = 2$ in all experiments). At each simulated timestep, the runtime complexity of an agent is linear in the number of neighbors [3]. Therefore, if an agent has n neighbors and needs $\mathcal{O}(n)$ time to simulate each one of them for each of the q actions, the runtime complexity of *C-Nav* is $\mathcal{O}(qn^2)$ per agent.

The complexity added by the one-way communication in *C-Nav* is negligible because a single message is communicated regardless of the number of neighbors. Therefore, this only requires a constant and small amount of resources (each message is formed by a 2D real number plus the ID of the agent). In time units, ORCA takes approx. 1.5×10^{-5} seconds to compute a new collision-free velocity, while *C-Nav* takes approx. 7×10^{-5} to evaluate and select a new preferred velocity for the next timestep. All in all, *C-Nav* takes approx. 8.5×10^{-5} of processing time for each agent, which allows us to simulate large scale navigation tasks in real time.

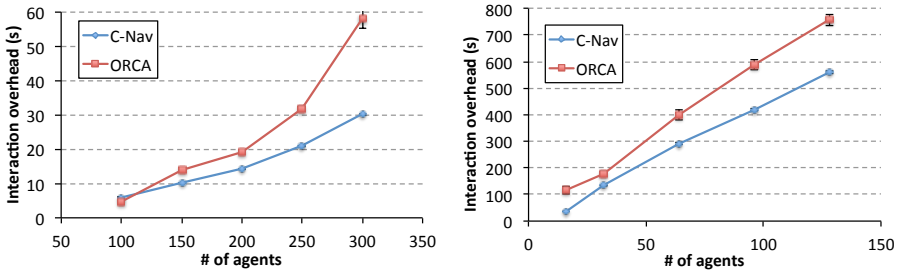
7.2. Scalability

We analyzed the scalability of our approach in the Crowd and Congested scenarios by varying the number of simulated agents and evaluating the interaction overhead time. The results, depicted in Figure 10 show that, in both scenarios, the overhead time of *C-Nav* increases more or less linearly as more agents are added, allowing us to simulate up to 300 agents in real time (in the Crowd scenario). At the same time, the overhead time introduced by each added agent in the system is not larger in our approach than in ORCA.

7.3. Effect of the coordination factor

We evaluated how the balance between the *goal-oriented* and the *constraint-reduction* components of our reward function (Eq. 2), controlled by the coordination-factor γ, affects the performance of *C-Nav* in all scenarios. Note that we did not test scenarios using a γ value of 1, as with this value the agents have no motivation to move towards their goal.

The results, shown in Figure 11, indicate that using values of γ smaller than 0.5 produces more contention in crowded situations, as agents prioritize their



(a) (b)

Figure 10: Interaction overhead of *C-Nav* and ORCA, in the (a) CROWD and (b) CONGESTED scenarios, with different number of agents.

own goal progress to their neighbors'. Overall, the best performance is achieved with $\gamma = 0.8$. This indicates that, for a time-efficient global navigation, agents should mainly aim at reducing the constraints of their neighbors. At the same time, they still need to account for their own goal progress, as the travel time starts increasing again in most scenarios with $\gamma > 0.8$. Two interesting cases can be observed in the CONGESTED and LINE scenarios (which share the same obstacle configuration), where the performance is improved when $\gamma = 0.9$. Here, all agents must go through the narrow exit to reach their goals (see Figure 5(c) and (d)). Consequently, as the value of γ approaches 1, agents are more likely to always defer to the agents closer to the single goal (regardless of the cost in terms of their own goal progress). In these scenarios, this extremely polite behavior creates an ordering between agents for passing through the narrow exit, as no two agents will defer to each other at the same time. Ultimately, this translates into better overall performance.

7.4. Effect of number of constrained neighbors

We also evaluated how the number of constrained neighbors, k, in the constraint-reduction component of the reward function (Eq. 4) affects the performance of our approach. Results for all scenarios can be observed in Figure 12. When k=0, the results are equivalent to the "selfish" *C-Nav* agents (see Figure 8).

We can observe that, in most scenarios, polite behavior towards even a single agent translates into a significant decrease in interaction overhead. In the CROWD scenario, though, this shift from 0 to 1 constrained neighbor (k) translates into an increase of interaction overhead, before reducing again when $k \geq 2$. In this scenario, the global benefit of deferring to a single neighbor is not enough to compensate for the extra travel time incurred by the polite agent. Only when agents are polite to groups of at least two neighbors it does make sense to "pay" the cost in terms of the travel time needed to reach their own goals.

In the BIDIRECTIONAL, CIRCLE, INTERSECTION and CROWD scenarios, the performance improves as each agent evaluates the constraints of more neighbors.

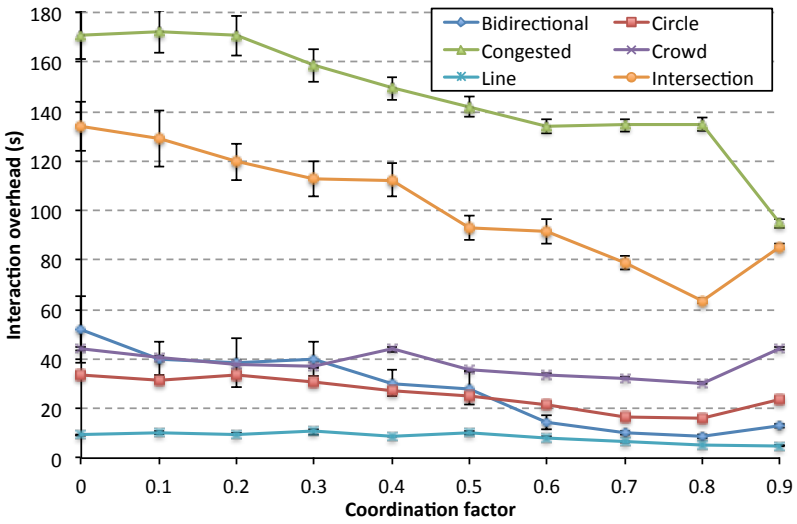


Figure 11: Performance comparison in all scenarios in *C-Nav*, with different values of the coordination factor. The error bars correspond to the standard deviation.

In these scenarios, as agents account for more neighbors upon computing a new velocity, their motion becomes more coordinated and the travel time of the entire system of agents is reduced (but the more neighbors each agent considers, the higher is the runtime complexity). However, adding more neighbors generally translates into smaller improvements. The reason is that as we add more neighbors, the influence of a specific neighbor's motion constraints in the reward function of the agent is reduced, as it is averaged with the other $k-1$ neighbors (Eq. 4). As such, the agent ends up prioritizing its own goal-progress versus the average progress of its k constrained neighbors. In all of the evaluated scenarios, we found the optimal threshold for k to be between 3 or 4 neighbors, with the exception of the CONGESTED scenario. Here, only one neighbor is needed to achieve the best results: considering more neighbors in this scenario translates into agents being very polite for a longer time (to allow agents in front to pass through the narrow doorway) which translates into larger delays in their own travel time. In all our simulation experiments, we used $k = 4$.

7.5. Effect of reduction of livelocks

In Section 5, we demonstrated that as the value of the coordination factor γ asymptotically approaches 1, the probability of livelocks with *C-Nav*, in environments with a single goal, reaches 0. Here, we evaluate how such relation translates into lower interaction overhead times. Specifically, we evaluated the interaction overhead in the CONGESTED scenario, with 32, 64 and 96 agents, as we increase the coordination factor in the range between 0.9 and 1. Results can

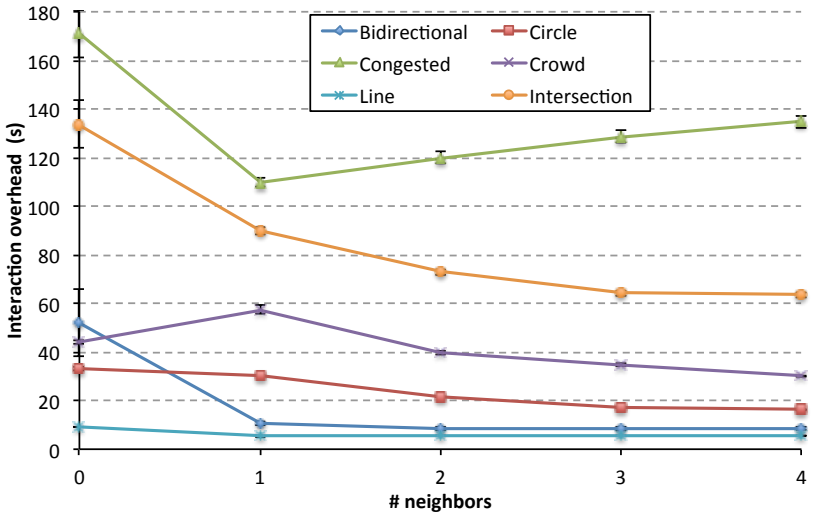


Figure 12: Performance comparison in all scenarios in *C-Nav*, with different number of neighbors considered in the politeness computation. The error bars correspond to the standard deviation.

be seen in Table 4.

In all three cases, we can observe that the interaction overhead values progressively decrease as the value of γ approaches 1. The higher the value of γ, the less likely is that agents impose constraints in the motions of neighbors that are closer to the (only) goal of the scenario, which helps increase the flow of agents through the narrow opening, reducing the overall travel time of the entire system of agents.

We also evaluated the difference between the preferred velocity, $\mathbf{v}^{\text{pref}}$, and the collision-free velocity, $\mathbf{v}^{\text{new}}$, for the agent that is, at each timestep, closest to the goal during the entire navigation task. The motion constraints imposed on this agent, unlike other agents farther from the goal, are a more clear indication of the effect of *C-Nav*'s polite behavior. The results, shown in Table 5, indicate that this value also decreases as γ asymptotically approaches 1 in all cases.

Although not directly measuring livelocks, these results indicate that with values of γ approaching 1, *C-Nav* agents lower the amount of motion constraints introduced into their neighbors, especially the agent closest to the goal. These results provide empirical evidence that supports the theoretical properties of *C-Nav*, described in Section 5.

8. C-Nav evaluation in multi-robot experiments

Here, we show how the advantages of accounting for polite behavior, as implemented in *C-Nav*, translate into multi-robot navigation tasks with non-holonomic constraints. To account for such constraints, we extended *C-Nav* to

Coord. factor	32 agents	64 agents	96 agents
0.9	90.73	231.01	365.68
0.95	74.64	183.85	304.55
0.99	66.1	157.32	254.82
0.999	65.18	155.6	244.51
0.9999	63.21	151.33	243.78

Table 4: Interaction overhead (in seconds) for *C-Nav* agents in the CONGESTED scenario, with 32, 64 and 96 agents, with values of the coordination factor approaching 1.

Coord. factor	32 agents	64 agents	96 agents
0.9	0.96	1.11	1.15
0.95	0.79	0.96	1.03
0.99	0.69	0.82	0.87
0.999	0.65	0.77	0.83
0.9999	0.65	0.76	0.82

Table 5: Difference between the preferred velocity, $\mathbf{v}^{\text{pref}}$, and the collision free velocity, $\mathbf{v}^{\text{new}}$, for the agent closest to the goal at each timestep, averaged through the duration of the navigation task in the CONGESTED scenario, with values of the coordination factor approaching 1.

the non-holonomic version of ORCA (NH-ORCA) [1], as implemented in [28] for ROS. We compared *C-Nav* to NH-ORCA in three real world navigation environments involving three Turtlebot 2 robots.

8.1. Experimental setup

We used ROS Indigo and Ubuntu 14.04 as the software platform for the robot experiments. In the ROS implementation of NH-ORCA [28], robots share their positions and collision-free velocities via ROS topics. We extended this implementation by allowing robots to also share their preferred velocities, as required by *C-Nav*. As the robots were equipped only with Kinect-type sensors that provide a narrow field of view, to allow them to accurately determine the positions and shapes of the obstacles, we hand-coded the obstacle positions in *C-Nav* ROS code (visible as black tape in Figures 16 and 17 and in the supplementary video). This reduced the source of potential perception errors in the navigation, and allowed the robots to focus on agent interactions.

We consider three real world scenarios, as depicted in Figure 13, involving three Turtlebot 2 robots as follows:

- CORRIDOR: Two robots cross paths with a single robot moving in the opposite direction in a narrow corridor bounded by virtually impassible obstacles (Fig 13(a)).

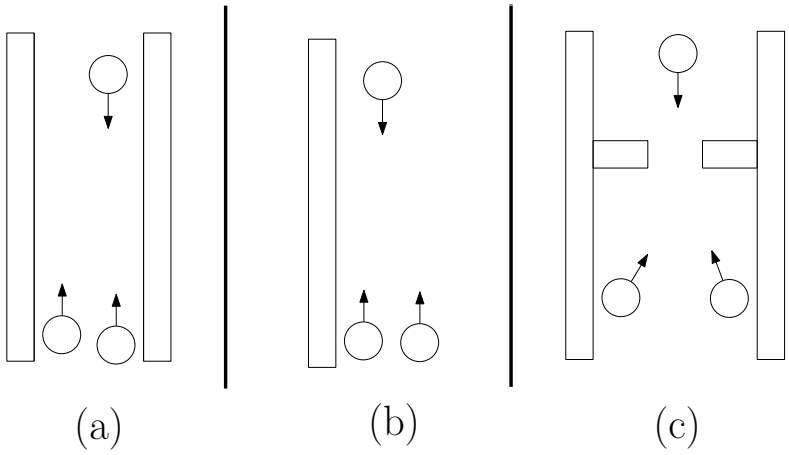


Figure 13: Three real world scenarios were used to evaluate *C-Nav* with robots: (a) Corridor, (b) 2vs1 and (c) SmallIntersection.

- 2vs1: Similar to the Corridor but with extra space for maneuvering in one side of the environment (Fig 13(b)).
- SmallIntersection: Two robots are located on one side of a narrow doorway and another robot on the other side, and all of them must travel to the other side of the doorway (Fig 13(c)).

All experiments were performed at the Applied Motion Lab in the Department of Computer Science at the University of Minnesota. Stills of the experiments are shown in Figures 16 and 17. The resulting robots' behaviors can be seen in the supplementary video.

8.2. Results

Tables 6 and 7 report quantitative results for all scenarios, in terms of the average travel time and maximum travel time of the robots, respectively. In the Corridor scenario, NH-ORCA was not able to move all robots to their goal positions: the two robots moving in the same direction pushed the third robot back, away from its goal (Figure 14(b)). When reaching their goals, these two robots block the way for the last robot, rendering it unable to reach its goal (see Figure 14(c) and the supplementary video). In the same scenario, *C-Nav* robots coordinate with each other in a distributed manner to reach their goals (Figure 14(d) and (e)).

In the 2vs1 scenario, both *C-Nav* and NH-ORCA robots were able to reach their goals, with *C-Nav* being considerably faster than NH-ORCA at this task. The sole NH-ORCA robot exhibits conservative behavior waiting for the other two robots to pass around it before taking any action and move forward resulting in large travel time. On the other hand, in the SmallIntersection scenario, the polite behavior displayed by *C-Nav* robots leads to a slight increase in travel

Method	Corridor	2vs1	SmallIntersection
NH-ORCA	∞	25	21
C-Nav	21	17	22

Table 6: Travel times (in seconds) for the last robot to reach its goal in the three scenarios of Figure 13, for NH-ORCA and *C-Nav*.

Method	Corridor	2vs1	SmallIntersection
NH-ORCA	∞	19	16
C-Nav	18.7	14.3	18

Table 7: Average travel times (in seconds) among all three robots, in the three scenarios of Figure 13, for NH-ORCA and *C-Nav*.

time, as compared to NH-ORCA (see Figures 15 and 17). Specifically, using NH-ORCA, the single robot attempting to reach the other side of the narrow area is forced to move aside to allow the two incoming robots to pass to the other side (Figure 15(b)), and can only resume its goal-oriented motion once the other robots are in their goal positions (Figure 15(c)). With *C-Nav*, the two robots make way for the single robot to pass through the doorway (Figure 15(d)), and move towards their goals only after the single robot is no longer in their path (Figure 15(e)).

9. Limitations of C-Nav

C-Nav has some limitations that arise from the nature of the interactions in crowded environments. As noted, agents are not always able to take actions that reduce the constraints of their neighbors (given that the agent itself might be constrained). A *C-Nav* agent makes optimistic action decisions by assuming that agents that are farther from its goal than itself will defer to its motion. As this is not always the case, and agents are often pushed against their intended motions, the positive effects of our approach can be limited in certain cases.

The constraint-reduction component of a *C-Nav* agent takes into account neighboring agents that are closer to the goal than itself. This is ideal for situations in which agents share a single goal (such as in the Congested scenario), but it might result in unnecessarily polite behavior in other cases, as agents will try to avoid imposing any constraints to each other. This behavior explains the slight increase in interaction overhead times for most scenarios where agents do not share the same goal (see $\gamma = 0.9$ in Figure 11).

Further, *C-Nav* agents do not make any distinction between their neighbors beyond the degree of constraints in their motion. This prevents *C-Nav* from identifying groups of agents moving in similar directions. To address this, agents could take into account the relation between their neighbors' intended velocities and adapt their politeness accordingly. For example, agents could

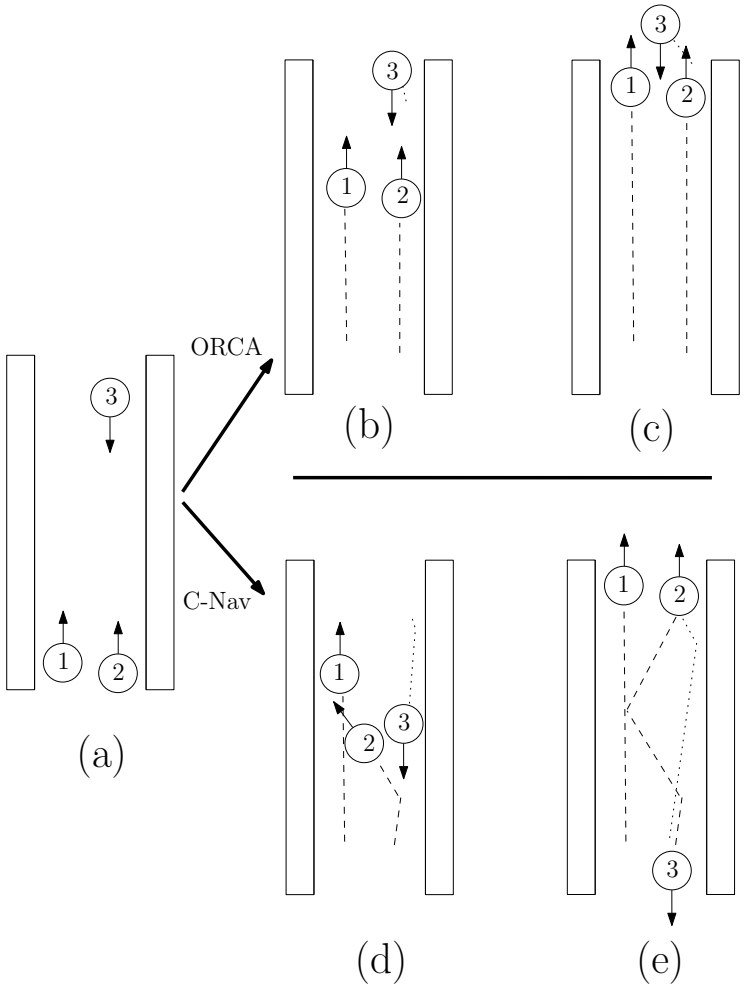


Figure 14: Trajectories of robots in the CORRIDOR scenario, with NH-ORCA and *C-Nav*: (a) Initial positions. (b) With NH-ORCA, robots 1 and 2 move to their goals pushing robot 3 away from its goal oriented path. (c) Final positions with NH-ORCA. Robots 1 and 2 reach their goals, resulting in robot 3 getting stuck, unable to reach its goal. (d) *C-Nav* robot 2 decides to follow robot 1, creating space for robot 3 to move to its goal. (e) Final positions with *C-Nav*. All robots are able to reach their goals.

behave more politely towards neighbors with similar intended velocities, than towards neighbors moving against them.

Finally, *C-Nav* assumes that all agents share the same underlying collision avoidance method. Should this not be the case, the predictions computed by the SimMotion procedure, when trying to assess how constrained is a neighbor in

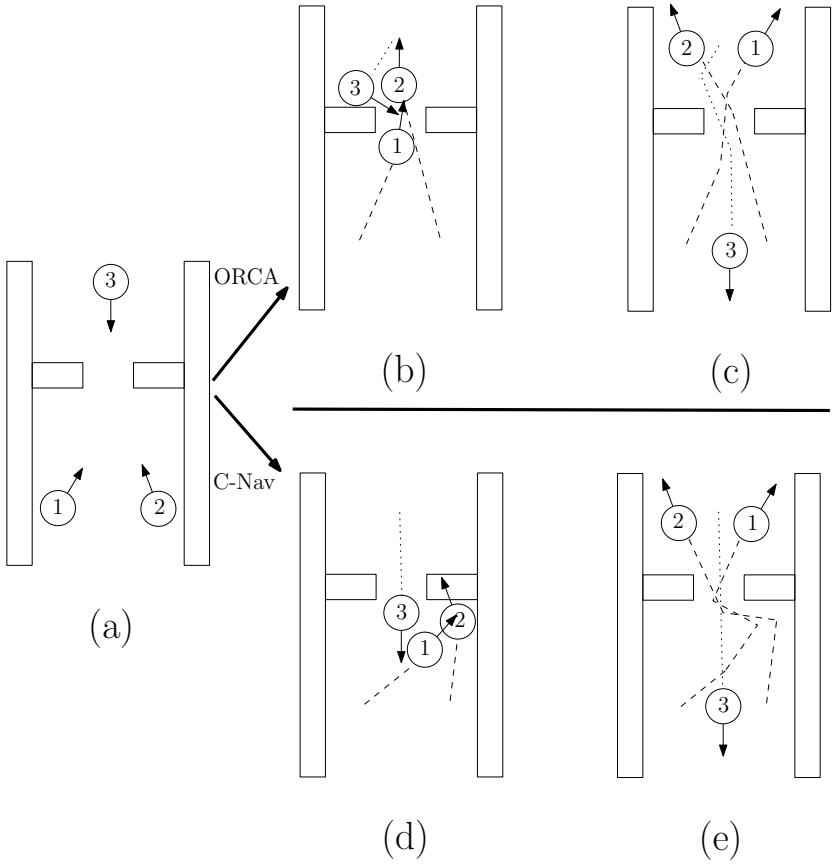
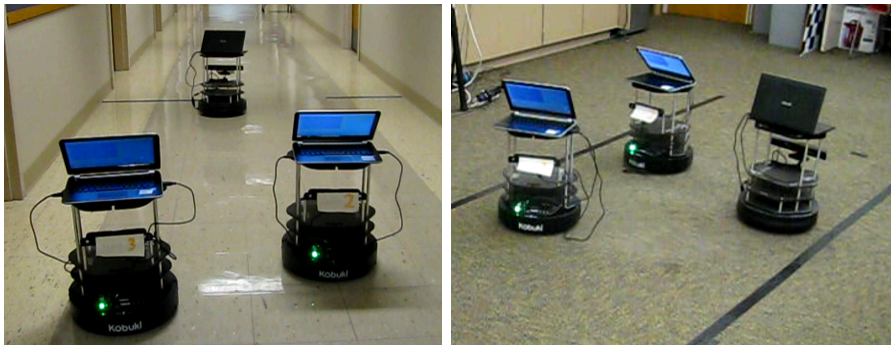


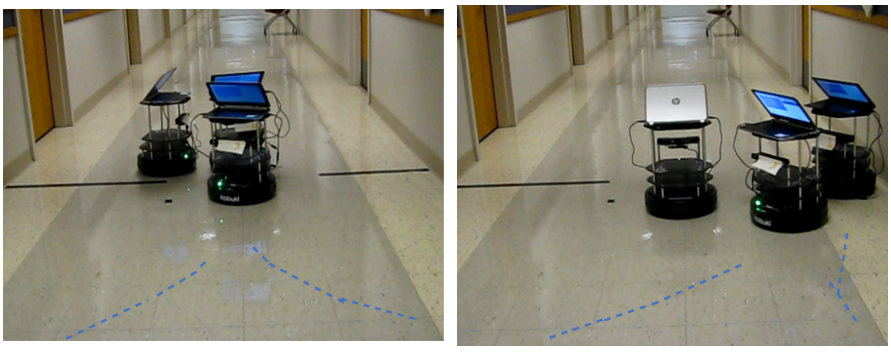
Figure 15: Trajectories of robots in the SmallIntersection scenario, with NH-ORCA and *C-Nav*: (a) Initial positions. (b) With NH-ORCA, robots 1 and 2 move to their goals, forcing robot 3 to wait before continuing its goal oriented motion. (c) Final positions with NH-ORCA, along with the robots' trajectories. (d) *C-Nav* robots 1 and 2 decide to make room for robot 3 to pass through the narrow opening. (e) Final positions with *C-Nav*, along with the robots' trajectories.

response to an action, might be inaccurate. *C-Nav* also assumes that agents can broadcast their intended velocities. If this is not the case (i.e., non-communicative agents), our approach would still work, though agents would only optimize their motions based on their own goal progress and the performance gain might not be as significant (as shown in Section 7). To address this limitation, an idea is to explore methods to predict the agents' preferred velocities from a sequence of observed velocities, such as in [18], or through deep learning methods, such as in [9].



(a) (b)

Figure 16: (a) Robots in the 2vs1 scenario. (b) Robots in the Corridor scenario. Black tape represents an impassible (virtual) obstacle.



(a) (b)

Figure 17: Different behaviors of robots using (a) NH-ORCA and (b) *C-Nav* in the SmallIntersection scenario. Black tape represents an impassible (virtual) obstacle.

10. Conclusions and Future Work

In this paper, we showed how introducing politeness into multi-agent navigation tasks results in more efficient and coordinated global motion. To do this we proposed *C-Nav*, a distributed coordination method for multi-agent navigation. In *C-Nav*, agents share their intended velocities via broadcast, and use this information (from their neighbors) to compute motions that reduce the constraints imposed on each other's intended motion, helping their entire neighborhoods to move closer to their goals. We prove that, in environments where the agents share a single goal, it is possible to guarantee the absence of livelocks as the coordination factor γ approaches 1.

We evaluated *C-Nav* in simulation in a variety of scenarios with different number of agents, showing that the distributed coordination achieved using *C-Nav* allows agents to reach their goals much faster and spending less energy than using only collision avoidance to plan their motions, or using a learning-based approach to select polite actions. We further evaluated *C-Nav* in multi-robot navigation tasks, demonstrating its advantage even if the agents are non-holonomic and subject to acceleration and kinematic constraints. *C-Nav* is highly scalable to hundreds of agents, and the communication method (broadcast) does not involve a significant extra computation to ORCA.

Looking forward, there are many possible avenues for future research. Currently, *C-Nav* agents do not distinguish between agents moving in a similar direction and agents moving in opposite directions. If taken into account, this distinction may allow agents to adapt their coordination strategy accordingly (for example, being more polite towards agents moving with them than towards them), which might increase the efficiency of their motions. Another interesting question is what components of an agent's state should be broadcast if we allow for limited one-way communication. Here, we assume that the intended velocity (i.e. the agent's preferred velocity) is the most important one, but perhaps there are other state and/or latent variables that can further improve coordination. Recent work in [29] may provide interesting insights towards answering this question. Finally, we are interested in combining *C-Nav* with a global planning approach, which might prove useful in environments where obstacles block the direct goal path for the agents (for example, in warehouse-like environments).

Acknowledgements

This work was partially funded by CONICYT under Grant FONDECYT INICIACION 11191197 and the University of Concepcion under Grant VRID INICIACION 218.093.018-1.0IN.

References

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

robots, in: Distributed Autonomous Robotic Systems. Springer, pp. 203–216.

[2] Bayazit, O., Lien, J.M., Amato, N., 2003. Better group behaviors in complex environments using global roadmaps, in: 8th International Conference on Artificial life, pp. 362–370.

[3] van den Berg, J., Guy, S.J., Lin, M., Manocha, D., 2011. Reciprocal n-body collision avoidance, in: Proc. International Symposium of Robotics Research. Springer, pp. 3–19.

[4] Cheng, H., Zhu, Q., Liu, Z., Xu, T., Lin, L., 2017. Decentralized navigation of multiple agents based on ORCA and model predictive control, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE. pp. 3446–3451.

[5] Cohen, L., Uras, T., Koenig, S., 2015. Feasibility study: using highways for bounded-suboptimal multi-agent path finding, in: Eighth Annual Symposium on Combinatorial Search.

[6] Curtis, S., Guy, S.J., Zafar, B., Manocha, D., 2011. Virtual Tawaf: A case study in simulating the behavior of dense, heterogeneous crowds, in: Proc. Workshop at Int. Conf. on Computer Vision, pp. 128–135.

[7] Ding, W., Li, S., Qian, H., Chen, Y., 2018. Hierarchical reinforcement learning framework towards multi-agent navigation, in: 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), IEEE. pp. 237–242.

[8] Everett, M., Chen, Y.F., How, J.P., 2018. Motion planning among dynamic, decision-making agents with deep reinforcement learning, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE. pp. 3052–3059.

[9] Fan, T., Long, P., Liu, W., Pan, J., 2018. Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios. arXiv preprint arXiv:1808.03841 .

[10] Fehr, E., Fischbacher, U., 2004. Social norms and human cooperation. Trends in cognitive sciences 8, 185–190.

[11] Fiorini, P., Shiller, Z., 1998. Motion planning in dynamic environments using Velocity Obstacles. The Int. J. of Robotics Research 17, 760–772.

[12] Fridman, N., Kaminka, G.A., 2010. Modeling pedestrian crowd behavior based on a cognitive model of social comparison theory. Computational and Mathematical Organization Theory 16, 348–372.

[13] Funge, J., Tu, X., Terzopoulos, D., 1999. Cognitive modeling: knowledge, reasoning and planning for intelligent characters, in: 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 29–38.

[14] Garcimartín, A., Pastor, J.M., Martín-Gómez, C., Parisi, D., Zuriguel, I., 2017. Pedestrian collective motion in competitive room evacuation. Scientific reports 7, 10792.

[15] Godoy, J., Chen, T., Guy, S.J., Karamouzas, I., Gini, M., 2018. ALAN: adaptive learning for multi-agent navigation. Autonomous Robots 42, 1543–1562.

[16] Godoy, J., Karamouzas, I., Guy, S.J., Gini, M., 2014. Anytime navigation with progressive hindsight optimization, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems.

[17] Godoy, J., Karamouzas, I., Guy, S.J., Gini, M., 2016a. Implicit coordination in crowded multi-agent navigation., in: Proc. AAAI Conf. on Artificial Intelligence.

[18] Godoy, J., Karamouzas, I., Guy, S.J., Gini, M., 2016b. Moving in a crowd: Safe and efficient navigation among heterogeneous agents., in: Proc. Int. Joint Conf. on Artificial Intelligence.

[19] Gupta, J.K., Egorov, M., Kochenderfer, M., 2017. Cooperative multi-agent control using deep reinforcement learning, in: International Conference on Autonomous Agents and Multiagent Systems, Springer. pp. 66–83.

[20] Guy, S., Chhugani, J., Kim, C., Satish, N., Lin, M., Manocha, D., Dubey, P., 2009a. Clearpath: highly parallel collision avoidance for multi-agent simulation, in: ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 177–187.

[21] Guy, S., Kim, S., Lin, M., Manocha, D., 2011. Simulating heterogeneous crowd behaviors using personality trait theory, in: Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 43–52.

[22] Guy, S.J., Chhugani, J., Curtis, S., Pradeep, D., Lin, M., Manocha, D., 2010. PLEdestrians: A least-effort approach to crowd simulation, in: ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 119–128.

[23] Guy, S.J., Chhugani, J., Kim, C., Satish, N., Lin, M., Manocha, D., Dubey, P., 2009b. Clearpath: highly parallel collision avoidance for multi-agent simulation, in: Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 177–187.

[24] Haghani, M., Sarvi, M., 2018. Crowd behaviour and motion: Empirical methods. Transportation research part B: methodological 107, 253–294.

[25] He, L., Pan, J., Wang, W., Manocha, D., 2016. Proxemic group behaviors using reciprocal multi-agent navigation, in: Proc. IEEE Int. Conf. on Robotics and Automation.

[26] Helbing, D., Farkas, I., Vicsek, T., 2000. Simulating dynamical features of escape panic. Nature 407, 487–490.

[27] Helbing, D., Molnar, P., 1995. Social force model for pedestrian dynamics. Physical review E 51, 4282.

[28] Hennes, D., Claes, D., Meeussen, W., Tuyls, K., 2012. Multi-robot collision avoidance with localization uncertainty, in: Proc. Int. Conf. on Autonomous Agents and Multi-Agent Systems.

[29] Hildreth, D., Guy, S.J., 2019. Coordinating multi-agent navigation by learning communication. Proceedings of the ACM on Computer Graphics and Interactive Techniques 2. doi:10.1145/3340261.

[30] Hönig, W., Kumar, T.S., Cohen, L., Ma, H., Xu, H., Ayanian, N., Koenig, S., 2016. Multi-agent path finding with kinematic constraints, in: Proc. Int'l Conf. on Automated Planning and Scheduling.

[31] Hoy, M., Matveev, A.S., Savkin, A.V., 2015. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. Robotica 33, 463–497.

[32] Jansen, M., Sturtevant, N., 2008. Direction maps for cooperative pathfinding, in: Proc. Artificial Intelligence and Interactive Digital Entertainment (AIIDE), pp. 185–190.

[33] Karamouzas, I., Guy, S.J., 2015. Prioritized group navigation with formation velocity obstacles, in: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 5983–5989.

[34] Karamouzas, I., Overmars, M., 2012. Simulating and evaluating the local behavior of small pedestrian groups. IEEE Trans. Vis. Comput. Graphics 18, 394–406.

[35] Khatib, O., 1986. Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robotics Research 5, 90–98.

[36] Knepper, R.A., Rus, D., 2012. Pedestrian-inspired sampling-based multi-robot collision avoidance, in: Proc. IEEE Int. Symp. on Robot and Human Interactive Communication, pp. 94–100.

[37] Koh, W.L., Zhou, S., 2011. Modeling and simulation of pedestrian behaviors in crowded places. ACM Transactions on Modeling and Computer Simulation (TOMACS) 21, 20.

[38] Kretzschmar, H., Spies, M., Sprunk, C., Burgard, W., 2016. Socially compliant mobile robot navigation via inverse reinforcement learning. The International Journal of Robotics Research 35, 1289–1307.

[39] Li, J., Harabor, D., Stuckey, P.J., Ma, H., Koenig, S., 2019. Disjoint splitting for multi-agent path finding with conflict-based search, in: Proceedings of the International Conference on Automated Planning and Scheduling, pp. 279–283.

[40] Lin, M.C., Sud, A., Van den Berg, J., Gayle, R., Curtis, S., Yeh, H., Guy, S., Andersen, E., Patil, S., Sewall, J., et al., 2008. Real-time path planning and navigation for multi-agent and crowd simulations, in: International Workshop on Motion in Games, Springer. pp. 23–32.

[41] Long, P., Liu, W., Pan, J., 2017. Deep-learned collision avoidance policy for distributed multiagent navigation. IEEE Robotics and Automation Letters 2, 656–663.

[42] Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O.P., Mordatch, I., 2017. Multi-agent actor-critic for mixed cooperative-competitive environments, in: Advances in Neural Information Processing Systems, pp. 6379–6390.

[43] Martinez-Gil, F., Lozano, M., Fernández, F., 2015. Strategies for simulating pedestrian navigation with multiple reinforcement learning agents. Autonomous Agents and Multi-Agent Systems 29, 98–130.

[44] Martinez-Gil, F., Lozano, M., Fernandez, F., 2017. Emergent behaviors and scalability for multi-agent reinforcement learning-based pedestrian models. Simulation Modelling Practice and Theory 74, 117–133.

[45] Masoud, S.A., Masoud, A.A., 2002. Motion planning in the presence of directional and regional avoidance constraints using nonlinear, anisotropic, harmonic potential fields: a physical metaphor. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 32, 705–723.

[46] Mavrogiannis, C.I., Thomason, W.B., Knepper, R.A., 2018. Social momentum: A framework for legible navigation in dynamic multi-agent environments, in: Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, pp. 361–369.

[47] Nicolas, A., Bouzat, S., Kuperman, M.N., 2017. Pedestrian flows through a narrow doorway: Effect of individual behaviours on the global flow and microscopic dynamics. Transportation Research Part B: Methodological 99, 30–43.

[48] Olfati-Saber, R., Fax, J.A., Murray, R.M., 2007. Consensus and cooperation in networked multi-agent systems. Proceedings of the IEEE 95, 215–233.

[49] Olfati-Saber, R., Murray, R.M., 2004. Consensus problems in networks of agents with switching topology and time-delays. IEEE Trans. Autom. Control 49, 1520–1533.

[50] Ondřej, J., Pettré, J., Olivier, A.H., Donikian, S., 2010. A synthetic-vision based steering approach for crowd simulation. ACM Trans. Graphics 29, 123.

[51] Park, J.J., Johnson, C., Kuipers, B., 2012. Robot navigation with model predictive equilibrium point control, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE. pp. 4945–4952.

[52] Pastor, J.M., Garcimartín, A., Gago, P.A., Peralta, J.P., Martín-Gómez, C., Ferrer, L.M., Maza, D., Parisi, D.R., Pugnaloni, L.A., Zuriguel, I., 2015. Experimental proof of faster-is-slower in systems of frictional particles flowing through constrictions. Physical Review E 92, 062817.

[53] Patil, S., Van den Berg, J., Curtis, S., Lin, M.C., Manocha, D., 2011. Directing crowd simulations using navigation fields. IEEE Trans. Vis. Comput. Graphics 17, 244–254.

[54] Pelechano, N., Allbeck, J., Badler, N., 2007. Controlling individual agents in high-density crowd simulation, in: Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 99–108.

[55] Pendleton, S.D., Andersen, H., Du, X., Shen, X., Meghjani, M., Eng, Y.H., Rus, D., Ang, M.H., 2017. Perception, planning, control, and coordination for autonomous vehicles. Machines 5, 6.

[56] Pentheny, G., 2015. Advanced techniques for robust, efficient crowds. Game AI Pro 2: Collected Wisdom of Game AI Professionals , 173.

[57] Popelová, M., Bída, M., Brom, C., Gemrot, J., Tomek, J., 2011. When a couple goes together: walk along steering, in: Motion in Games, Springer. pp. 278–289.

[58] Reynolds, C.W., 1987. Flocks, herds and schools: A distributed behavioral model. ACM SIGGRAPH Computer Graphics 21, 25–34.

[59] Reynolds, C.W., 1999. Steering behaviors for autonomous characters, in: Game Developers Conference, pp. 763–782.

[60] Rimon, E., Koditschek, D.E., 1992. Exact robot navigation using artificial potential functions. Departmental Papers (ESE) , 323.

[61] Shao, W., Terzopoulos, D., 2007. Autonomous pedestrians. Graphical Models 69, 246–274.

[62] Siméon, T., Leroy, S., Lauumond, J.P., 2002. Path coordination for multiple mobile robots: A resolution-complete algorithm. IEEE Transactions on Robotics and Automation 18, 42–49.

[63] Sisbot, E.A., Marin-Urias, L.F., Alami, R., Simeon, T., 2007. A human aware mobile robot motion planner. IEEE Transactions on Robotics 23, 874–883.

[64] Sud, A., Andersen, E., Curtis, S., Lin, M.C., Manocha, D., 2008. Real-time path planning in dynamic virtual environments using multiagent navigation graphs. IEEE transactions on visualization and computer graphics 14, 526–538.

[65] Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., Vicente, R., 2017. Multiagent cooperation and competition with deep reinforcement learning. PloS one 12, e0172395.

[66] Trautman, P., Ma, J., Murray, R.M., Krause, A., 2015. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. The Int. J. of Robotics Research 34, 335–356.

[67] Yu, C., Zhang, M., Ren, F., Luo, X., 2013. Emergence of social norms through collective learning in networked agent societies, in: Proc. Int. Conf. on Autonomous Agents and Multi-Agent Systems, pp. 475–482.

[68] Zhang, J., Klingsch, W., Schadschneider, A., Seyfried, A., 2012. Ordering in bidirectional pedestrian flows and its influence on the fundamental diagram. Journal of Statistical Mechanics: Theory and Experiment 2012, P02002.