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

Julio Godoy and **Ioannis Karamouzas** and **Stephen J. Guy** and **Maria Gini**
Department of Computer Science and Engineering
University of Minnesota
200 Union St SE, Minneapolis MN 55455
{godoy, ioannis, sjguy, gini}@cs.umn.edu

Abstract

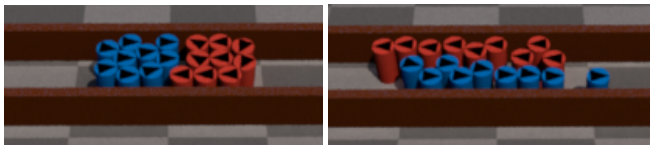
In crowded multi-agent navigation, the motion of the agents is significantly constrained by the motion of the nearby agents. This makes planning paths very difficult and leads to inefficient global motion. To address this problem, we recently proposed a distributed approach to implicitly coordinate the motions of agents in crowded environments. With our approach, agents take into account the velocities and goals of their neighbors and optimize their motion accordingly and in real-time. In this paper, we extend our approach by performing a more thorough theoretical analysis, and experimentally demonstrate its robustness to different actions, different types of information broadcasted and a larger variety of scenarios.

1 Introduction

Decentralized navigation of multiple agents in crowded environments has application in many domains such as swarm robotics, traffic engineering and crowd simulation. 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 the agents need to reach their destinations in a timely manner while still guaranteeing a collision-free motion.

A variety of approaches have been proposed to address this problem. Recently, velocity-based approaches have gained popularity due to their robustness and their ability to provide collision-free guarantees about 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 can result in globally inefficient behavior, long travel times and in worst case, deadlocks.

Recently, in [Godoy *et al.*, 2016], we showed that by accounting for the intended motion of its neighbors, an agent can choose a velocity that improves the time-efficiency of the entire crowd, reducing the time when all agents reach their goals (see Fig. 1).



(a) ORCA (b) C-Nav

Figure 1: Two groups of 9 agents each move 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.

To that end, we proposed *C-Nav* (short for Coordinated Navigation), a distributed approach where agents improve the global motion of a crowd by implicitly coordinating their local motions. This coordination is achieved using observations of the nearby agents' motion patterns and a limited one-way communication, allowing *C-Nav* to scale up to 300 agents. 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. In this paper, we extend the *C-Nav* work by introducing new features such as an increased set of actions for the agents and different types of one-way communication. We also perform additional experiments and discuss some theoretical properties of *C-Nav*.

2 Related Work

Multi-Agent Navigation. In the last two decades, a number of models have been proposed to simulate the motion of agents. At a broad level, these models can be classified into flow-based and agent-based methods. Flow-based methods focus on the behavior of the crowd as a whole and dynamically compute vector-fields to guide its motion [Treuille *et al.*, 2006; Narain *et al.*, 2009]. Even though such approaches can simulate interactions of dense crowds, due to their centralized nature, they are prohibitively expensive when there are many agents with distinct goals. In contrast, agent-based models are purely decentralized and plan for each agent independently. After the seminal work of Reynolds on *boids* [1987], many agent-based approaches have been introduced, including social forces [Helbing and Molnar, 1995], psychological models [Pelechano *et al.*, 2007] as well as behavioral and cognitive models [Shao and Terzopoulos, 2007]. How-

ever, the majority of such agent-based techniques do 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 [Fiorini and Shiller, 1998] have been proposed that compute collision-free velocities for the agents using either sampling [Ondřej *et al.*, 2010; Karamouzas and Overmars, 2012] or optimization-based techniques [van den Berg *et al.*, 2011; Guy *et al.*, 2009]. In particular, the Optimal Reciprocal Collision Avoidance navigation framework, ORCA [van den Berg *et al.*, 2011], plans provably collision-free velocities for the agents and has been successfully applied to simulate high-density crowds [Curtis *et al.*, 2011]. 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 implicitly coordinate their motions in order to improve the global time-efficiency of the crowd.

Coordination in Multi-Agent Navigation. Many approaches have been proposed to allow agents to coordinate their motions while moving to their goals. In [Jansen and Sturtevant, 2008], 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 [Pentheny, 2015]. Other approaches consider the generation of bounded suboptimal paths for multiple agents [Cohen *et al.*, 2015] and the kinematic constraints that emerge when dealing with embodied agents [Hönig *et al.*, 2016]. However, 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 [Patil *et al.*, 2011], where agents can be guided to their goals to avoid congestion. Our approach does not need external guidance and automatically generates coordinated goal paths for the agents.

When simulating the motion of pedestrians, coordination can be achieved through social norms [Fehr and Fischbacher, 2004], which can be embedded in the system [Koh and Zhou, 2011] or can emerge through the interactions between the agents [Yu *et al.*, 2013]. Other works such as [Fridman and Kaminka, 2010] assume similar behavior between the agents and use cognitive models of social comparison. Similarly, our work allows agents to compare motion features, but no socio-psychological theory is used. Another approach for achieving coordinated navigation is by explicitly forming groups of agents, and have the groups follow specific directions while maintaining cohesiveness among their members [Karamouzas and Overmars, 2012; Karamouzas and Guy, 2015]. A recent work addresses emergent group formation of different sizes based on proxemics, but it cannot simulate merging or splitting of the groups [He *et al.*, 2016].

Multi-agent coordination can also be achieved using learning methods, such as in [Melo and Veloso, 2011], which learn the value of joint actions when coordination is required, and use Q-learning when it is not. The approach in of [Martinez-Gil *et al.*, 2014] uses reinforcement learning for simulating pedestrian navigation, allowing the agents to learn policies offline that can then be applied to specific scenarios. To the contrary, Godoy et al. [2015] use online learning where agents adapt their motions with no communication or need for offline training. However, all these approaches consider only up to 40 agents, while we aim at having up to 300 agents, as in the method we present here.

3 Problem Formulation

In our problem setting, there are n independent agents $A_1 \dots A_n$, each with an individual start and goal position. For simplicity, we assume that the 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 $v_i^{\max}$. Furthermore, A_i has a set of 9 empirically defined preferred velocities $\mathbf{v}_i^{\text{pref}}$ (see Fig. 2a) that indicate the agent's desired direction and speed of motion at a given timestep. A_i also has a goal velocity $\mathbf{v}_i^{\text{goal}}$ directed toward the agent's goal $\mathbf{g}_i$ with a magnitude equal to $v_i^{\max}$. In the absence of any other agents and static obstacles, the preferred velocity of A_i, $\mathbf{v}_i^{\text{pref}}$, is equal to its goal velocity $\mathbf{v}_i^{\text{goal}}$. We assume that an agent can sense the radii, positions and 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.

Definition 1. *Intended Velocity: A design choice, this velocity can correspond either to the agent's (1) preferred velocity, or to (2) its goal-oriented velocity, based on its short term and long term desired motion, respectively.*

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 and with the environment, while reaching their goals as fast as possible. More formally, we seek to minimize the arrival time of the last agent or equivalently the maximum travel time of the agents while guaranteeing collision-free motions. This objective is similar to minimizing the makespan of the travel time of all agents. 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 to find for each agent a new velocity that respects its geometric and kinematics constraints while progressing the agent towards its goal.

To obtain a collision-free velocity, we use the ORCA navigation framework [van den Berg *et al.*, 2011]. In each timestep, ORCA takes as input a preferred velocity $\mathbf{v}^{\text{pref}}$ and returns a new velocity $\mathbf{v}^{\text{new}}$ that is collision-free and as close as possible to $\mathbf{v}^{\text{pref}}$, 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 typically have only a goal-oriented $\mathbf{v}^{\text{pref}}$, they may get stuck in local minima, which leads to large travel times and, subsequently, globally inefficient motions.

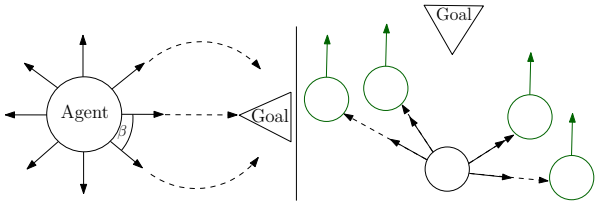


Figure 2: The two sets of actions in *C-Nav*, where the dotted lines indicate the projection of the corresponding preferred velocities. (a) Agent-based actions: moving at 1.5 m/s with different angles with respect to the goal: $0°$, β, $-\beta$, $90°$, $-90°$, $180°$, $180° + \beta$, $180° - \beta$ and complete stop. In our implementation, $\beta = 45°$ (b) Following actions: follow a specific neighbor agent at maximum speed.

To address the aforementioned issues we propose *C-Nav*, an implicit coordination approach that allows each agent to choose from two sets of preferred velocities at each timestep (Fig. 2), with the objective of optimizing both its own goal progress as well as the progress of its neighbors.

4 The *C-Nav* Approach

In *C-Nav*, the agents use information about the intended motion of their neighbors to make better decisions on how to move and implicitly coordinate with each other. This reduces the travel time of all the agents.

Agents using *C-Nav* can select from a set of actions (i.e., preferred velocities) to decide how to move in order to reduce the travel time of all agents. We classify these actions into (1) individual agent-based actions (Fig. 2a) and (2) following actions (Fig. 2b). Throughout this work, we use the term action and preferred velocity interchangeably.

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 0.1 seconds (line 3). In each new update, the agent computes which of its neighbors move in a similar manner as itself (line 4) and which neighbors are most constrained in their motions (line 5), and uses this information to evaluate all of its actions (line 7). After this evaluation, the best action is selected (line 9). Finally, its intended velocity $\mathbf{v}^{\text{int}}$ is broadcasted to the agent's neighbors (line 10) and its preferred velocity is 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) and the cycle repeats.

Algorithm 1: The *C-Nav* framework for an agent

1: start the navigation
2: **while** not at the goal **do**
3: **if** $UpdateAction(t)$ **then**
4: compute most similar agents
5: compute most constrained agents
6: **for all** $a \in Actions$ **do**
7: $\mathcal{R}_a \leftarrow SimMotion(a)$
8: **end for**
9: $\mathbf{v}^{\text{pref}} \leftarrow \arg\max_{a \in Actions} \mathcal{R}_a$
10: broadcast ID and $\mathbf{v}^{\text{int}}$ to nearby agents
11: **end if**
12: $\mathbf{v}^{\text{new}} \leftarrow CollisionAvoidance(\mathbf{v}^{\text{pref}})$
13: $\mathbf{p}^{t} \leftarrow \mathbf{p}^{t-1} + \mathbf{v}^{\text{new}} \cdot \Delta t$
14: **end while**

4.1 Agent neighborhood information

With information obtained by sensing (radii, positions and velocities) and via one-way communication (IDs and intended velocity) from all the neighbors within the sensing range, each agent estimates the most similar nearby agents and the most constrained ones.

Motion similarity

Each agent evaluates the similarity between the motion of its neighbors and its own (see Algorithm 2), identifying those neighbors that are moving faster than itself and in a similar direction. By following such neighbors, the time-efficiency of all agents can be increased.

To evaluate this similarity, the agents follows a two-step process (Algorithm 2). Firstly, each agent filters those neighbors whose intended velocity is not in the direction of the agent's goal (line 4). Secondly, the agent determines which of its neighbors with similar intended motions are moving faster than the agent to its goal (line 5), by projecting the neighbor's observed velocity to the agent's goal oriented vector. The similarity value for each neighbor j, $SimVal_j$, measures how closely related are the agent's goal vector and the neighbor's current observed velocity. Finally, these similarity values are sorted in descending order (line 8) and the corresponding list of the neighbors' indices is returned.

Algorithm 2: Compute most similar neighbors of i

1: **Input:** list of neighbors $\mathcal{N}(i)$
2: **Output:** Sim_{rank}, list of indices of the most similar neighbors
3: **for all** $j \in \mathcal{N}(i)$ **do**
4: **if** $\mathbf{v}_j^{\text{int}} \cdot \frac{\mathbf{g}_i - \mathbf{p}_i}{\|\mathbf{g}_i - \mathbf{p}_i\|} > 0$ **then**
5: $SimVal_j \leftarrow \mathbf{v}_j^{\text{new}} \cdot \frac{\mathbf{g}_i - \mathbf{p}_i}{\|\mathbf{g}_i - \mathbf{p}_i\|}$
6: **end if**
7: **end for**
8: $Sim_{rank} \leftarrow Sort(SimVal)$

Following actions. Once an agent knows how similar is the motion of its neighbors, it can choose a velocity at maximum speed towards one of them (Fig. 2b). As such, agents can follow each other, leading to the emergent formation of lanes that allow them to efficiently move to their goals (Fig. 1b).

Constrained neighborhood motion

Agents can also use the intended velocities of their neighbors to evaluate how constrained is their motion and, thus, determine neighbors that are more likely to slow down the overall progress of the crowd. By reducing the constraints of these neighbors, the time-efficiency of the system increases.

Algorithm 3: Compute most constrained neighbors of i

1: **Input:** list of neighbors $\mathcal{N}(i)$
2: **Output:** C_{rank}, list of indices of the most constrained neighbors
3: **for all** $j \in \mathcal{N}(i)$ **do**
4: **if** $\|\mathbf{g}_i - \mathbf{p}_j\| < \|\mathbf{g}_i - \mathbf{p}_i\|$ **then**
5: $C_j \leftarrow \| \mathbf{v}_j^{\mathrm{int}} - \mathbf{v}_j^{\mathrm{new}} \|$
6: **end if**
7: **end for**
8: $C_{rank} \leftarrow Sort(C)$

Algorithm 3 evaluates how constrained is the motion of each neighbor. To do this, each agent compares a neighbor's intended motion with its observed velocity (line 5). The larger the difference, the more likely it is that its 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 4). This ensures that no two agents with the same goal will simultaneously defer to each other. The agent keeps a list C which quantifies the constraints of each neighbor. After all neighbors have been evaluated, C is sorted in descending order, and a list C_{rank} of the indices of the sorted neighbors is returned (line 8).

Once the agent computes a ranking of the most similar and 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 in an efficient manner while, on the other, help nearby agents to move according to their intended motions.

5 Action Evaluation and Selection

Agents can choose a preferred velocity from the two sets of actions shown in Figure 2. The first set allows agents to choose velocities that are uniformly distributed in the space of directions. The second set, on the other hand, allows agents to execute the following behaviors, described in Section 4.1, with up to s neighbors ($0 \leq s \leq |\mathcal{N}|$). To evaluate each 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 corresponds to $SimMotion(a)$ in Algorithm 1 (line 7).

5.1 Motion simulation

As a first step, an agent simulates the changes in its neighborhood, updating the velocities and positions of itself and its neighbors for each timestep within a fixed time horizon T, for each action. Note that in very crowded areas, 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. This prevents the agent from selecting a velocity that improves the motion of its most constrained neighbors. In *C-Nav* the agent considers in its simulation only the neighbors that are closer to its goal than itself, 'ignoring' the agents that are behind it. 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.

5.2 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 goal. Our reward function balances these two objectives, by taking a linear combination of a *goal-oriented*, and a *constrained-reduction* component (Eq. 1). Each component has an upper bound of 1 and a lower bound of -1 and is weighted by the *coordination-factor* γ.

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

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

$$\mathcal{R}_a^g = \frac{\sum_{t=0}^{T-1} \left(\mathbf{v}_i^{\mathrm{new}} \cdot \frac{\mathbf{g}_i - \mathbf{p}_i}{\|\mathbf{g}_i - \mathbf{p}_i\|} \right)}{T \cdot v_i^{\max}} \tag{2}$$

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 introduce constraints into these k agents. More formally:

$$\mathcal{R}_a^c = \frac{\sum_{t=1}^{T-1} \sum_{j \in C_{rank}} \left(v_i^{\max} - \| \mathbf{v}_j^{\mathrm{int}} - \mathbf{v}_j^{\mathrm{new}} \| \right)}{(T-1) \cdot k \cdot v_i^{\max}} \tag{3}$$

If an agent only aims at maximizing $\mathcal{R}_a^g$, its behavior 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 implicitly coordinates its goal-oriented motion with that of its neighbors, resulting in lower travel times for all agents.

6 Theoretical Analysis

We focus our 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 deferent motions that move the agents away from their goals, which would prevent the progress of the agents. We show that the probability of livelocks occurring approaches 0 as the value of γ (Eq. 1) asymptotically approaches 1, in scenarios where all agents share the same goal (e.g., the Congested scenario in Fig. 3).

For the purpose of this analysis, assume that after an agent reaches the goal, it is removed from the environment. Assume also that A_α corresponds 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 5). 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 on the value of the action's goal progress $\mathcal{R}_a^g$. To ensure that A_α has 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 $A_\alpha \in \mathcal{N}(i)$) can choose an action a' that moves it backwards from its goal, that does not introduce 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 5), 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 $\mathcal{R}_a^g$, the probability that A_i will choose action a' approaches 1 as $\gamma \to 1$ (see Eq. 1). As each agent A_i chooses a', it will not introduce constraints into A_α's motion, which will be able to move as if it was the only agent in the system, and it will be able 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. *The probability of livelocks occurring in C-Nav, in environments with a single goal, approaches 0 as γ asymptotically approaches 1. Under these conditions, the probability that all agents reach the goal approaches 1.*

7 Experiments

We evaluated *C-Nav* on five scenarios (see Fig. 3 for their layout) using a varying number of agents. Each result corresponds to the average over 30 simulations (see `http://motion.cs.umn.edu/r/CNAV/` for videos), as ORCA introduces randomness in the preferred velocities. The scenarios are as follows:

- **Bidirectional:** 18 agents are clustered in two groups that move to the opposite side of a narrow corridor formed by two walls.
- **Circle:** 128 agents are placed along the circumference of a circle and must reach their antipodal positions.
- **Crowd:** 300 agents are randomly placed in a densely populated area and are given random goals.

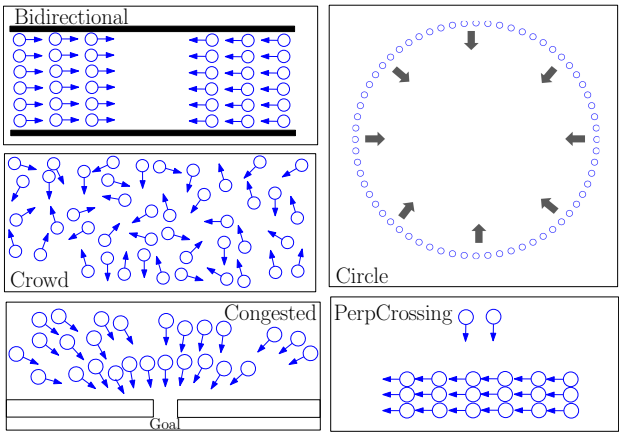


Figure 3: Layout of the scenarios used to evaluate *C-Nav*. We consider scenarios with up to 300 agents and, in some cases, with static obstacles.

- **Congested:** 32 agents are placed very close to the narrow exit of an open hallway and must escape the hallway through this exit.
- **PerpCrossing:** Two agents have orthogonally intersecting paths with a crowd of 24 agents.

To evaluate our approach, we measure the time that the agents spent in order to resolve interactions with each other and the environment. We estimate this time by computing the difference between the maximum travel time of the agents and the hypothetical travel time if agents were able to just follow their shortest paths to their goals at full speed:

$$\max_i(TimeToGoal(A_i)) - \max_i\left(\frac{shortestPath(A_i)}{v_i^{\max}}\right)$$

We call this metric *interaction overhead*. A theoretical property of this metric is that an interaction overhead of 0 represents a lower bound on the optimal travel time for the agents, and it is the best result that an optimal centralized approach could potentially achieve.

Using the interaction overhead metric, we compared *C-Nav* to vanilla ORCA and the ALAN approach of Godoy et al. [2015]. In all our experiments, we used ORCA's default settings for the agent's radii (0.5 m), sensing range (15 m) and maximum number of agents sensed ($|\mathcal{N}|$=10). We set T=2 timesteps and $v^{\max}$=1.5 m/s. We empirically set γ to 0.8 and both k and s=3, as they produced the best performance overall. The timestep duration, Δt, was also empirically set to 25 ms, as it provided each agent a fast reaction time to the changes in the velocities of other agents. All simulations ran in real time for all evaluated methods.

7.1 Results

Figure 4 shows the interaction overhead of the three methods in all scenarios, where, in *C-Nav*, agents only communicate their preferred velocities. The interaction overhead of *C-Nav* is significantly lower than ORCA and ALAN in all cases, which indicates that by considering information about their neighborhood, agents can coordinate their motions and

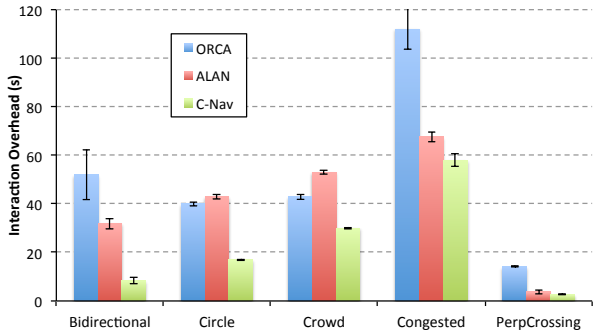


Figure 4: Performance comparison between ORCA, ALAN and the *C-Nav* approach. In all scenarios, agents using our coordination approach have the lowest overhead times. The error bars correspond to the standard error of the mean.

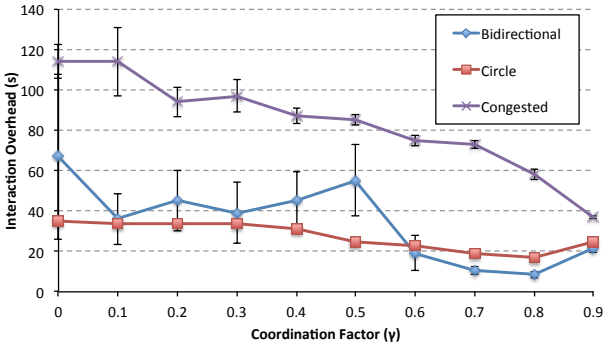


Figure 6: Performance of *C-Nav* agents in the Bidirectional, Circle and Congested scenarios, with different values of the coordination-factor γ, where agents communicate their preferred velocities.

improve their time-efficiency. We observe emergent behavior in the Bidirectional and Circle scenario, where agents going in the same direction form lanes. Such lanes reduce the constraints in other agents leading to more efficient simulations.

It is worth noting that agents using ALAN outperform ORCA agents in the Bidirectional, Congested and PerpCrossing scenarios. However, the unnecessary exploration performed by ALAN agents prevents this method from scaling to more than 100 agents (as in the Circle and Crowd scenarios), where ALAN cannot outperform ORCA.

We also evaluated the interaction overhead of *C-Nav* agents with different types of intended velocities $\mathbf{v}^{\text{int}}$ (see Section 4): the preferred velocity ($\mathbf{v}^{\text{pref}}$) or the goal velocity ($\mathbf{v}^{\text{goal}}$), along with the case of no communication ($None$). The results in Figure 5 show that, in all cases, the communication of intended motions helps agents improve their time-efficiency as compared to the absence of communication. Overall, using the preferred velocity as the intended motion achieves the best performance. The only exception is in the Congested scenario, where the goal path is constrained by static obstacles, which means that deferring to the goal velocity of the neighbors is more critical than in other scenarios.

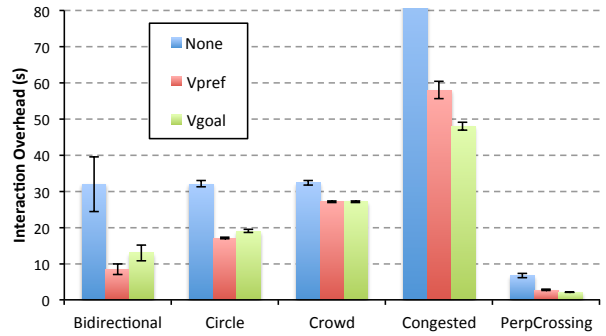


Figure 5: Performance comparison in *C-Nav* where agents communicate: no information ($None$), only their preferred velocities ($\mathbf{v}^{\text{pref}}$) or only their goal velocities ($\mathbf{v}^{\text{goal}}$).

Effect of the coordination-factor (γ). We evaluated how the balance between the *goal-oriented* and the *constraint reduction* components of our reward function (Eq. 1) controlled by the coordination-factor γ, affects the performance of *C-Nav* in three scenarios. We can observe in Figure 6 that using values of $\gamma < 0.5$ has a more negative impact on performance as compared to larger values ($\gamma > 0.5$). This indicates that accounting for the neighbors' intended velocities improves the time-efficiency for all agents. In the Congested scenario, higher γ values correlate with lower travel times, which follows the result from Theorem 1. In the Circle and Bidirectional scenarios, the best performance is achieved with γ =0.8, while larger values show slightly increase in the travel time. Note that we do not include an evaluation for $\gamma = 1$, as in this case the agents have no incentive to reach their goals. Overall, *C-Nav* shows good performance with $\gamma > 0.5$.

8 Conclusions and Future Work

We have extended *C-Nav*, a recently proposed coordination approach for large scale multi-agent systems. *C-Nav* agents use their sensing input and a limited one-way communication to implicitly coordinate their motions. In this paper, we introduced an increased set of actions for the agents and different types of one-way communication. We also performed additional experiments and discuss some theoretical properties of *C-Nav*. We show that *C-Nav* performs well with different types of intended motions communicated and is robust to a wide variety of scenarios.

Our approach does not limit the acceleration of the agents, which might produce non-smooth motions in simulations. Further, we assume that agents can broadcast their intended velocities. If this is not the case, results indicate high interaction overhead times. To address this limitation, we would like to explore methods to predict the agents' preferred velocities from a sequence of observed velocities, using, e.g., a hidden Markov model.

Acknowledgment: Support for this work is gratefully acknowledged from the University of Minnesota Informatics Institute, and by the NSF through grants #CNS-1544887 and #CHS-1526693.

References

[Cohen *et al.*, 2015] Liron Cohen, Tansel Uras, and Sven Koenig. Feasibility study: using highways for bounded-suboptimal multi-agent path finding. In *Eighth Annual Symposium on Combinatorial Search*, 2015.

[Curtis *et al.*, 2011] Sean Curtis, Stephen J Guy, Basim Zafar, and Dinesh Manocha. Virtual tawaf: A case study in simulating the behavior of dense, heterogeneous crowds. In *Proc. Workshop at Int. Conf. on Computer Vision*, pages 128–135, 2011.

[Fehr and Fischbacher, 2004] Ernst Fehr and Urs Fischbacher. Social norms and human cooperation. *Trends in cognitive sciences*, 8(4):185–190, 2004.

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

[Fridman and Kaminka, 2010] Natalie Fridman and Gal A Kaminka. Modeling pedestrian crowd behavior based on a cognitive model of social comparison theory. *Computational and Mathematical Organization Theory*, 16(4):348–372, 2010.

[Godoy *et al.*, 2015] Julio Godoy, Ioannis Karamouzas, Stephen J. Guy, and Maria Gini. Adaptive learning for multi-agent navigation. In *Proc. Int. Conf. on Autonomous Agents and Multi-Agent Systems*, pages 1577–1585, 2015.

[Godoy *et al.*, 2016] Julio Godoy, Ioannis Karamouzas, Stephen J. Guy, and Maria Gini. Implicit coordination in crowded multi-agent navigation. In *Proc. AAAI Conference on Artificial Intelligence*, 2016.

[Guy *et al.*, 2009] Stephen J Guy, Jatin Chhugani, Changkyu Kim, Nadathur Satish, Ming Lin, Dinesh Manocha, and Pradeep Dubey. Clearpath: highly parallel collision avoidance for multi-agent simulation. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 177–187, 2009.

[He *et al.*, 2016] Liang He, Jia Pan, Wenping Wang, and Dinesh Manocha. Proxemic group behaviors using reciprocal multi-agent navigation. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 2016.

[Helbing and Molnar, 1995] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[Hönig *et al.*, 2016] Wolfgang Hönig, TK Satish Kumar, Liron Cohen, Hang Ma, Hong Xu, Nora Ayanian, and Sven Koenig. Multi-agent path finding with kinematic constraints. In *International Conference on Automated Planning and Scheduling*, 2016.

[Jansen and Sturtevant, 2008] M.R. Jansen and N.R. Sturtevant. Direction maps for cooperative pathfinding. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 185–190, 2008.

[Karamouzas and Guy, 2015] Ioannis Karamouzas and Stephen J Guy. Prioritized group navigation with formation velocity obstacles. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 5983–5989, 2015.

[Karamouzas and Overmars, 2012] Ioannis Karamouzas and Mark Overmars. Simulating and evaluating the local behavior of small pedestrian groups. *IEEE Trans. Vis. Comput. Graphics*, 18(3):394–406, 2012.

[Koh and Zhou, 2011] Wee Lit Koh and Suiping Zhou. Modeling and simulation of pedestrian behaviors in crowded places. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 21(3):20, 2011.

[Martinez-Gil *et al.*, 2014] Francisco Martinez-Gil, Miguel Lozano, and Fernando Fernández. MARL-Ped: A multi-agent reinforcement learning based framework to simulate pedestrian groups. *Simulation Modelling Practice and Theory*, 47:259–275, 2014.

[Melo and Veloso, 2011] Francisco S Melo and Manuela Veloso. Decentralized MDPs with sparse interactions. *Artificial Intelligence*, 175(11):1757–1789, 2011.

[Narain *et al.*, 2009] Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C Lin. Aggregate dynamics for dense crowd simulation. *ACM Trans. Graphics*, 28(5):122, 2009.

[Ondřej *et al.*, 2010] Jan Ondřej, Julien Pettré, Anne-Hélène Olivier, and Stéphane Donikian. A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graphics*, 29(4):123, 2010.

[Patil *et al.*, 2011] Sachin Patil, Jur Van den Berg, Sean Curtis, Ming C Lin, and Dinesh Manocha. Directing crowd simulations using navigation fields. *IEEE Trans. Vis. Comput. Graphics*, 17(2):244–254, 2011.

[Pelechano *et al.*, 2007] N. Pelechano, J.M. Allbeck, and N.I. Badler. Controlling individual agents in high-density crowd simulation. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 99–108, 2007.

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

[Reynolds, 1987] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM Siggraph Computer Graphics*, 21(4):25–34, 1987.

[Shao and Terzopoulos, 2007] W. Shao and D. Terzopoulos. Autonomous pedestrians. *Graphical Models*, 69(5-6):246–274, 2007.

[Treuille *et al.*, 2006] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. *ACM Trans. Graphics*, 25(3):1160–1168, 2006.

[van den Berg *et al.*, 2011] Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Proc. International Symposium of Robotics Research*, pages 3–19. Springer, 2011.

[Yu *et al.*, 2013] Chao Yu, Minjie Zhang, Fenghui Ren, and Xudong Luo. Emergence of social norms through collective learning in networked agent societies. In *Proc. Int. Conf. on Autonomous Agents and Multi-Agent Systems*, pages 475–482, 2013.