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

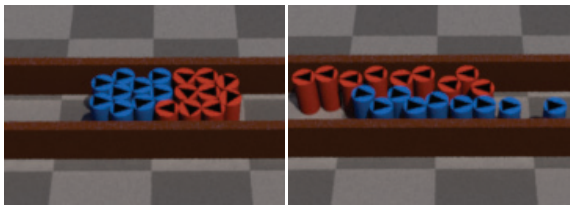
In crowded multi-agent navigation environments, the motion of the agents is significantly constrained by the motion of the nearby agents. This makes planning paths very difficult and leads to inefficient global motion. To address this problem, we propose a new distributed approach to coordinate the motions 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. We experimentally validate our coordination approach in a variety of scenarios and show that its performance scales to scenarios with hundreds of agents.

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 and unrealistic behavior, long travel times and in worst case, deadlocks.

Ideally, to reach globally efficient motions, a centralized entity needs to compute the best velocity for each agent. In a decentralized domain, where agents have only partial observation of the environment, such entity is not present. Agents can only use their limited knowledge of the environment to compute their local motions. In this paper, we hypothesize that if an agent could account for the intended motion of its nearby agents, then it could choose a velocity in a way that benefits not only itself but also its neighboring agents. Consider, for example, the two groups of agents in Figure 1a. Here, two groups of agents try to move past each other in a narrow hallway. The agents navigate using a predictive collision-avoidance technique, but still end up getting stuck in congestion. To address this, we seek to develop a navigation method that encourages coordination to emerge through agents' interactions. We accomplish this, by allowing agents to account for their neighbors' intended velocities during their planning. Figure 1b shows an example of such coordinated motion.



(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.

Following this idea, we propose *C-Nav* (short for Coordinated Navigation), a distributed approach to improve the global motion of a set of agents in crowded environments 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 to hundreds of 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.

This work makes three main contributions. First, we propose a framework for multi-agent coordination that leads to time-efficient motions. Second, we show that accounting for nearby agents when selecting an optimal velocity promotes implicit coordination between agents. Third, we experimentally validate our approach via simulations and show that it

leads to more coordinated and efficient motions in a variety of scenarios as compared to a state-of-the-art collision avoidance framework (van den Berg et al. 2011), and a recently proposed learning approach for multi-agent navigation (Godoy et al. 2015).

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 methods and agent-based approaches. Flow-based methods focus on the behavior of the crowd as a whole and dynamically compute vector-fields to guide its motion (Treuille, Cooper, and Popović 2006; Narain et al. 2009). Even though such approaches can simulate interactions of dense crowds, due to their centralized nature, they are prohibitively expensive for planning the movements of a large number of agents that have 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, Allbeck, and Badler 2007) as well as behavioral and cognitive models (Shao and Terzopoulos 2007). However, 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; Moussaïd, Helbing, and Theraulaz 2011; 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; Kim et al. 2012). 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 Systems. The coordination of multiple agents sharing a common environment has been widely studied in the Artificial Intelligence community. When communication is not constrained, coordination can be achieved by casting it as a distributed constraint optimization problem, where agents send messages back and forth until a solution is found (Modi et al. 2003; Ottens and Faltings 2008). Guestrin et al. (2002) assume that agents can directly communicate with each other by using coordination graphs, which also rely on a message-passing system. 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.

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 approaches of (Martinez-Gil, Lozano, and Fernández 2014; Torrey 2010) use reinforcement learning for multi-agent navigation, allowing the agents to learn policies offline that can then be applied to specific scenarios. These works consider a few agents (up to 40), while our work focuses on environments with hundreds of interacting agents. More recently, Godoy et al. (2015) use an online learning approach for adapting the motions of multiple agents with no communication without the need for offline training. However, the resulting behavior of this approach does not scale well to hundreds of agents, as opposed to the technique we present here.

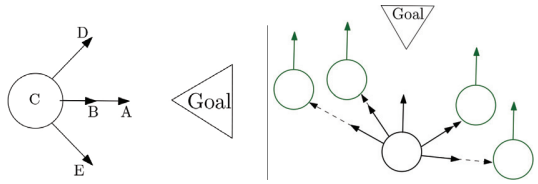
Problem Formulation

In our problem setting, there are n independent agents $A_1 \dots A_n$, each with a unique start and goal position. For simplicity, we assume that the agents move on a 2D plane where static obstacles O, approximated as line segments, can also be present. We model each agent A_i as a disc with a fixed radius r_i. At time 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 preferred velocity $\mathbf{v}_i^{\text{pref}}$ (commonly directed toward the agent's goal $\mathbf{g}_i$ with a magnitude equal to $v_i^{\max}$). We assume that an agent can sense the radii, positions and velocities of at most $numNeighs$ 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 preferred velocity. This type of communication scales well, as it is not affected by the size of the agent's neighborhood, unlike two-way communication whose complexity increases proportionally to the number of neighbors.

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:

$$\begin{aligned} \text{minimize} \quad & \max_i(TimeToGoal(A_i)) \\ \text{s.t.} \quad & \|\mathbf{p}_i^t - \mathbf{p}_j^t\| > r_i + r_j, \underset{i \neq j}{\forall} i, j \in [1, n] \\ & dist(\mathbf{p}_i^t, O_j) > r_i, \quad \forall i \in [1, n], j = 1 \dots |O| \\ & \|\mathbf{v}_i^t\| \leq v_i^{\max}, \quad \forall i \in [1, n] \end{aligned} \tag{1}$$

where $TimeToGoal(A_i)$ is the travel time of agent A_i from its start position to its goal and $dist(\cdot)$ denotes the Euclidean distance. Since the agents navigate *independently* with only limited communication, Eq. 1 has to be solved in a decentralized manner. Therefore, at each time instant, we seek to find for each agent a new velocity that respects the agent's



(a) (b)

Figure 2: Actions. (a) Agent-based actions: A. Towards the goal at maximum speed, B. towards the goal at reduced speed, C. stop, D. and E. towards the goal at a fixed angle. (b) Neighborhood-based actions: follow a specific neighbor agent or the goal-oriented motion at maximum speed.

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). 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.

To address the aforementioned issues, in addition to the set of preferred velocities defined in (Godoy et al. 2015) (Figure 2a), we also consider velocities with respect to certain key neighbors (Figure 2b). This creates an implicit coordination between the agents and enables our approach to compute globally efficient paths for the agents as well as scale to hundreds of agents. The ALAN framework (Godoy et al. 2015) achieves good performance in several environments, but the absence of communication limits the ability of agents to coordinate their motions.

The *C-Nav* Approach

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

Algorithm 1 outlines *C-Nav*. For each agent that has not reached its goal, we compute a new action, i.e., preferred velocity, 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 and which neighbors are most constrained in their motions (line 4 and line 5, respectively), and uses this information to evaluate all of its actions (line 7). After this evaluation, the best action is selected (line 9). Finally, this preferred velocity $\mathbf{v}^{\text{pref}}$ is broadcasted to the agent's neighbors (line 10) and 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 agent i

1: initialize simulation
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: $R_a \leftarrow SimMotion(a)$
8: **end for**
9: $\mathbf{v}^{\text{pref}} \leftarrow \arg\max_{a \in Actions} R_a$
10: broadcast ID and $\mathbf{v}^{\text{pref}}$ to nearby agents
11: **end if**
12: $\mathbf{v}^{\text{new}} \leftarrow CollisionAvoidance(\mathbf{v}^{\text{pref}})$
13: $\mathbf{p}^t \leftarrow \mathbf{p}^{t\text{-}1} + \mathbf{v}^{\text{new}} \cdot \Delta t$
14: **end while**

Agent neighborhood information

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

Neighborhood-based actions. Each agent i computes how similar the motions of its neighbors are to its own motion (see Algorithm 2). This allows the agent to locate neighbors that are moving faster than itself and in a similar direction. By following such neighbors, the time-efficiency of the agent can be increased. Specifically, the preferred velocities of the nearby agents are used to first select neighbors which goals in the same direction as the agent i (line 4). The actual velocity of each of these neighbors is then compared to i's goal-oriented vector to quantify the similarity between the agents (line 5). Algorithm 2 sorts these similarity values in a descending order and returns the corresponding list of the neighbors' indices.

Algorithm 2: Compute most similar neighbors of i

1: **Input:** list of neighbors $Neighs(i)$
2: **Output:** Sim_{rank}, list of indices of the most similar neighbors
3: **for all** $j \in Neighs(i)$ **do**
4: **if** $\mathbf{v}_j^{\text{pref}} \cdot \frac{\mathbf{g}_i - \mathbf{p}_i}{\|\mathbf{g}_i - \mathbf{p}_i\|} > 0$ **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: $Sim_{rank} \leftarrow Sort(SimVal)$

Once an agent knows which of its nearby agents have a similar motion, it can use this information to choose a velocity with maximum speed towards one of these neighbors (Figure 2b), in addition to the goal-oriented motion. These actions, unlike the ones in Figure 2a, do not have a fixed angle with respect to the agent's goal, as they depend on the position of individual neighbors.

Constrained neighborhood motion. Each agent i uses Algorithm 3 to evaluate how constrained the motions of its neighbors are and, thus, determine agents that are more likely to slow down the overall progress of the crowd. By yielding to those constrained neighbors, the global time-efficiency of the system increases. To avoid circular dependencies which can give rise to deadlocks, the agent i 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. We estimate how constrained a neighbor is based on the difference between its preferred and actual velocity (line 5). The larger the difference, the more likely it is that its motion is impeded. The agent keeps a list C with the evaluation of how constrained is the motion of each of its neighbors. After all neighbors have been evaluated, Algorithm 3 sorts C in descending order, and returns a list C_{rank} of the indices of the sorted neighbors (line 8). The agent uses this information when evaluating each of its available actions.

Algorithm 3: Compute most constrained neighbors of i

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

Action evaluation and selection

Agents can choose a preferred velocity from two sets of actions, an agent-based (Figure 2a) and a neighborhood-based (Figure 2b) action set. In the latter, agents consider up to s neighbors ($0 \leq s \leq numNeighs$). To estimate the fitness of the actions, the agent simulates each action 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 ($0 \leq k \leq numNeighs$). Algorithm 4 outlines this procedure.

Motion simulation. As a first step towards the selection of an action, an agent simulates the evolution of its neighborhood dynamics (line 4), that is, it updates the velocities and positions of itself and its neighbors for each timestep within a given time horizon T (line 3), for each possible action. Two timesteps is the minimum time horizon required to observe the effect of the agent's chosen motion.

It should be noted here that in very crowded areas, agents often have no control over their motions, as they are being pushed by other agents in order to avoid collisions. Hence, simulating the dynamics of all the agent's neighbors often results in the same velocity for all simulated actions. This does not help, because the agent would not be able to select a velocity that improves the motion of its most constrained neighbors. Therefore, the agent considers in its simulation

Algorithm 4: SimMotion(a) for agent i

1: **Input:** integer $a \in Actions$, list of neighbors $Neighs(i)$
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** $j \in Neighs(i)$ **do**
7: **if** $rank(j \in C_{rank}) < k$ **then**
8: $\mathcal{R}_a^c \leftarrow \mathcal{R}_a^c + v_i^{\max} - \| \mathbf{v}_j^{\text{pref}} - \mathbf{v}_j^{\text{new}} \|$
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: $\mathcal{R}_a^g \leftarrow \frac{\mathcal{R}_a^g}{T \cdot v_i^{\max}}, \mathcal{R}_a^c \leftarrow \frac{\mathcal{R}_a^c}{(T-1) \cdot k \cdot v_i^{\max}}$
15: $\mathcal{R}_a \leftarrow (1-\gamma) \cdot \mathcal{R}_a^g + \gamma \cdot \mathcal{R}_a^c$

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.

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. It computes this based on the difference between j's predicted collision-free velocity $\mathbf{v}_j^{\text{new}}$ and its communicated preferred velocity $\mathbf{v}_j^{\text{pref}}$ (line 8).

Motion evaluation. To decide what motion to perform next, the agent aims at minimizing the amount of constraints imposed by 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. 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 $\mathbf{v}^{\text{new}}$ 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:

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

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 introduce constraints into these k agents.

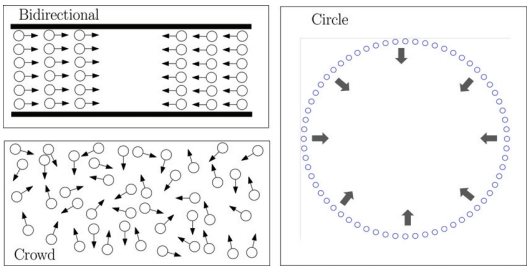


Figure 3: Scenarios. *Bidirectional*: two groups of agents move to the opposite side of a narrow corridor. *Crowd*: agents' initial and goal positions are placed randomly in a small area. *Circle*: agents move to their antipodal positions.

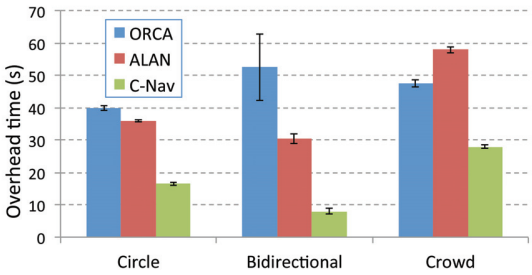


Figure 4: Performance comparison between ORCA, ALAN and our *C-Nav* approach. In all scenarios, agents using our coordination approach have the lowest overhead times. The error bars correspond to the standard error of the mean.

More formally:

$$\mathcal{R}_a^c = \frac{\sum_{t=1}^{T-1} \sum_{j \in C_{rank}} \left(v_i^{\max} - \| \mathbf{v}_j^{\text{pref}} - \mathbf{v}_j^{\text{new}} \| \right)}{(T-1) \cdot k \cdot v_i^{\max}} \quad (4)$$

If an agent only aims at maximizing $\mathcal{R}_a^g$, its behavior would be selfish and it would not consider the constraints that its actions impose on its most constrained 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.

Experiments

We evaluated *C-Nav* on three different scenarios using a varying number of agents (see Figure 3). Each result corresponds to the average over 30 simulations (see http://motion.cs.umn.edu/r/CNAV/ for videos). The scenarios are as follows:

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

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.

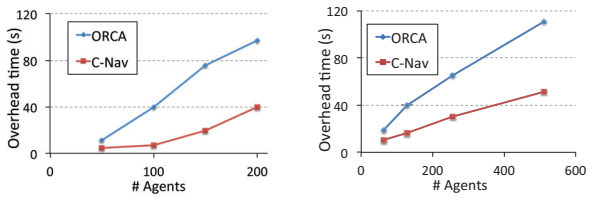
Using the interaction overhead metric, we compared *C-Nav* to vanilla ORCA (greedy goal-oriented motion) and the ALAN approach of Godoy et al. (2015). In all our experiments, we used ORCA's default settings for the agent's radii (0.5 m), sensing range (15 m) and maximum number of agents sensed ($numNeighs$=10). We set T=2 timesteps, $v^{\max}$=1.5 m/s, γ=0.9, k=3 and s=3. The timestep duration, Δt, is set to 25 ms. The number of timesteps T was chosen because it is the minimum needed to observe the effects of the motion and it produces the best results, though even with larger values our approach still outperforms both ALAN and ORCA. All simulations ran in real time for all evaluated methods.

Results

We evaluated the interaction overhead time in the Circle scenario with 128 agents, the Bidirectional scenario with 18 agents, and the Crowd scenario with 300 agents. Results can be seen in Figure 4. Agents using ALAN outperform ORCA agents in the Bidirectional scenario, and are on par with them in the Circle scenario. However, ALAN's performance fails to scale to 300 agents in the Crowd scenario. The interaction overhead of *C-Nav* is lower than ORCA and ALAN in all cases, which indicates that by considering information about their neighborhood, agents can coordinate their motion and improve their time-efficiency. In terms of qualitative results, 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.

Scalability. We analyzed the scalability of our approach in the Bidirectional and Circle scenarios by varying the number of simulated agents. The results are depicted in Figure 5. In both scenarios, the difference between the overhead times of ORCA and *C-Nav* increases as more agents are added. However, the overhead time introduced by each added agent in the system is lower in our approach than in ORCA.

Action sets. We evaluated how the use of only agent-based or neighborhood-based action sets compares to the combined action set that *C-Nav* employs. The results for the



(a) Bidirectional (b) Circle

Figure 5: Scalability results in the Bidirectional and Circle scenarios, in terms of interaction overhead time. In Bidirectional, the number of agents varied from 50 to 200. In the Circle, the number of agents varies from 64 to 512.

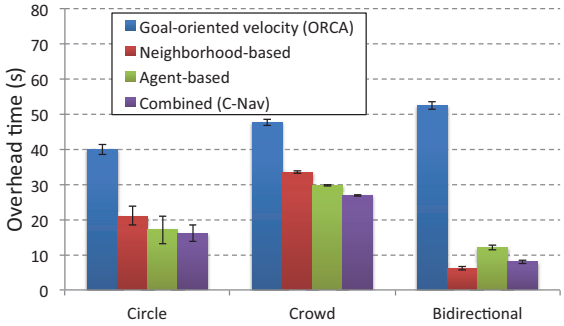


Figure 6: Performance of the four action selection methods in terms of interaction overhead time, for three scenarios.

three scenarios are shown in Figure 6. For completeness, we also report the performance of ORCA, i.e., the overhead time when only a single goal-oriented action is used. We can observe that the combined set of actions is either better or no worse than using just the neighborhood-based or the agent-based action set. The only exception is in the Bidirectional scenario, where the neighborhood-based set outperforms the combined one (the difference in overhead time is statistically significant). With only neighborhood-based actions, an agent will deviate from its goal-oriented velocity only when it can follow a neighbor which is already moving in a less constrained manner towards the agent's goal.

Effect of the coordination-factor (γ). We evaluated how the balance between the goal-oriented and the constrained-reduction components of our reward function (Eq. 2) controlled by the coordination-factor γ, affects the performance of our *C-Nav* approach. We can observe in Figure 7 that using any of the two extremes of γ, i.e., either a pure goal-oriented reward (γ=0) or a pure constrained-reduction reward (γ=1), can reduce the performance. Accounting only for goal-oriented behavior forces agents to choose velocities that in many cases prevent other agents from moving to their goals, reducing the overall time-efficiency in their navigation. On the other hand, giving only preference to reducing the other agents' constraints does not promote progress to the goal in low-density environments. Because *C-Nav* agents consider only neighbors that are likely to slow down the global motion, a high value of the coordination-factor

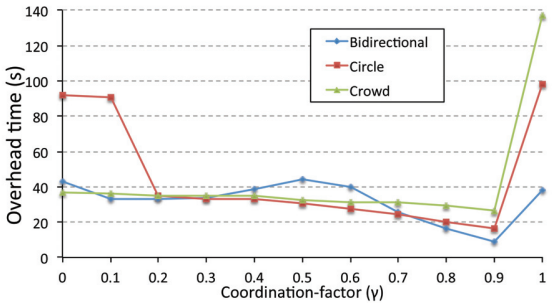


Figure 7: Performance of *C-Nav* agents, with different values of the coordination-factor γ.

(γ=0.9) helps both the agent and its neighbors to move to their respective goals, which results in the best performance.

Effect of number of constrained neighbors k. We also evaluated how the number of constrained neighbors (k) in the constrained-reduction component of the optimization function (Eq. 4) affects the performance of our approach. In general, the interaction overhead time decreases while k increases. 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. In the Crowd scenario, the interaction overhead time decreases nearly linearly as k increases, while in the other two scenarios the overhead time decreases exponentially with increasing k. We note, though, that in all of our experiments, considering more than 3 neighbors does not lead to any significant performance improvement.

Conclusions and Future Work

We have proposed *C-Nav*, a 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. Each agent takes advantage of the motion patterns of its nearby neighbors to avoid introducing constraints in their motions, and temporarily follow other agents that have similar motion. By doing this, agents in dense environments are able to reach their goals faster than using a state-of-the-art collision-avoidance framework and an adaptive learning approach for multi-agent navigation.

Our implementation assumes that agents can broadcast their preferred velocities. If this is not the case (i.e. non-communicative agents), *C-Nav* would still work, though agents would only optimize their motions based on their own goal progress. 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. Adapting *C-Nav* to physical robots moving in human populated environments is another exciting avenue for future work. The recent work of (Choi, Kim, and Oh 2014) and (Trautman et al. 2015) can provide some interesting ideas in this direction.

Acknowledgment: Support for this work is gratefully acknowledged from the University of Minnesota Informatics Institute.

References

Choi, S.; Kim, E.; and Oh, S. 2014. Real-time navigation in crowded dynamic environments using gaussian process motion control. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 3221–3226.

Curtis, S.; Guy, S. J.; Zafar, B.; and 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*, 128–135.

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

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

Godoy, J. E.; Karamouzas, I.; Guy, S. J.; and Gini, M. 2015. Adaptive learning for multi-agent navigation. In *Proc. Int. Conf. on Autonomous Agents and Multi-Agent Systems*, 1577–1585.

Guestrin, C.; Venkataraman, S.; and Koller, D. 2002. Context-specific multiagent coordination and planning with factored MDPs. In *Proc. AAAI Conference on Artificial Intelligence*, 253–259.

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

Helbing, D., and Molnar, P. 1995. Social force model for pedestrian dynamics. *Physical review E* 51(5):4282.

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

Kim, S.; Guy, S. J.; Manocha, D.; and Lin, M. C. 2012. Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory. In *Proc. of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 55–62.

Martinez-Gil, F.; Lozano, M.; and Fernández, F. 2014. Marlped: A multi-agent reinforcement learning based framework to simulate pedestrian groups. *Simulation Modelling Practice and Theory* 47:259–275.

Melo, F. S., and Veloso, M. 2011. Decentralized MDPs with sparse interactions. *Artificial Intelligence* 175(11):1757–1789.

Modi, P. J.; Shen, W.-M.; Tambe, M.; and Yokoo, M. 2003. An asynchronous complete method for distributed constraint optimization. In *Proc. Int. Conf. on Autonomous Agents and Multi-Agent Systems*, volume 3, 161–168.

Moussaïd, M.; Helbing, D.; and Theraulaz, G. 2011. How simple rules determine pedestrian behavior and crowd disasters. *Proc. of the National Academy of Sciences* 108(17):6884–6888.

Narain, R.; Golas, A.; Curtis, S.; and Lin, M. C. 2009. Aggregate dynamics for dense crowd simulation. *ACM Trans. Graphics* 28(5):122.

Ondřej, J.; Pettré, J.; Olivier, A.-H.; and Donikian, S. 2010. A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graphics* 29(4):123.

Ottens, B., and Faltings, B. 2008. Coordinating agent plans through distributed constraint optimization. In *Proc. of the ICAPS-08 Workshop on Multiagent Planning*.

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

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

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

Torrey, L. 2010. Crowd simulation via multi-agent reinforcement learning. In *Proc. Artificial Intelligence and Interactive Digital Entertainment*, 89–94.

Trautman, P.; Ma, J.; Murray, R. M.; and 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(3):335–356.

Treuille, A.; Cooper, S.; and Popović, Z. 2006. Continuum crowds. *ACM Trans. Graphics* 25(3):1160–1168.

van den Berg, J.; Guy, S. J.; Lin, M.; and Manocha, D. 2011. Reciprocal n-body collision avoidance. In *Proc. International Symposium of Robotics Research*. Springer. 3–19.