Coordinating Multi-Agent Navigation by Learning Communication

DALTON HILDRETH, University of Minnesota
STEPHEN J. GUY, University of Minnesota

This work presents a decentralized multi-agent navigation approach that allows agents to coordinate their motion through local communication. Our approach allows agents to develop their own emergent language of communication through an optimization process that simultaneously determines *what* agents say in response to their spatial observations and *how* agents interpret communication from others to update their motion. We apply our communication approach together with the TTC-Forces crowd simulation algorithm (a recent, high performing, anticipatory collision technique) and show a significant decrease in congestion and bottle-necking of agents, especially in scenarios where agents benefit from close coordination. In addition to reaching their goals faster, agents using our approach show coordinated behaviors including greeting, flocking, following, and grouping. Furthermore, we observe that communication strategies optimized for one scenario often continue to provide time-efficient, coordinated motion between agents when applied to different scenarios. This suggests that the agents are learning to generalize strategies for coordination through their communication "language".

CCS Concepts: • **Computing methodologies** → **Animation**; **Cooperation and coordination**; *Learning latent representations*; *Multi-agent systems*; *Multi-agent planning*.

Additional Key Words and Phrases: multi-agent communication, crowd simulation, local motion planning, learning for animation, language discovery

ACM Reference Format:
Dalton Hildreth and Stephen J. Guy. 2019. Coordinating Multi-Agent Navigation by Learning Communication. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2, Article 20 (July 2019), 17 pages. https://doi.org/10.1145/3340261

1 INTRODUCTION

Capturing the efficient, coordinated behavior common in human motion is important to a number of fields such as architecture, video games, movies, and virtual reality. Whether for simulations by architects to design buildings that provide better flows to dense crowds or for computer games with scores of moving characters, allowing simulated agents to navigate efficiently in a shared space is an important aspect of providing believable, natural, and socially coherent motion for virtual characters.

While there has been much recent work on how simulated agents can make independent decisions in the process of navigation, not as much is known about how to best allow agents to communicate with each other in ways that can improve the efficiency of navigation. Agents who are able to "talk" to their nearby neighbors should, in theory, be able to exchange critical information needed to better coordinate their motion and avoid potential bottlenecks, deadlocks, and other sources of

Authors' addresses: Dalton Hildreth, hildr039@umn.edu, University of Minnesota, 4-192 Keller Hall, 200 Union Street SE, Minneapolis, Minnesota, 55455; Stephen J. Guy, sjguy@umn.edu, University of Minnesota, 4-192 Keller Hall, 200 Union Street SE, Minneapolis, Minnesota, 55455.

2577-6193/2019/7-ART20 $15.00
https://doi.org/10.1145/3340261

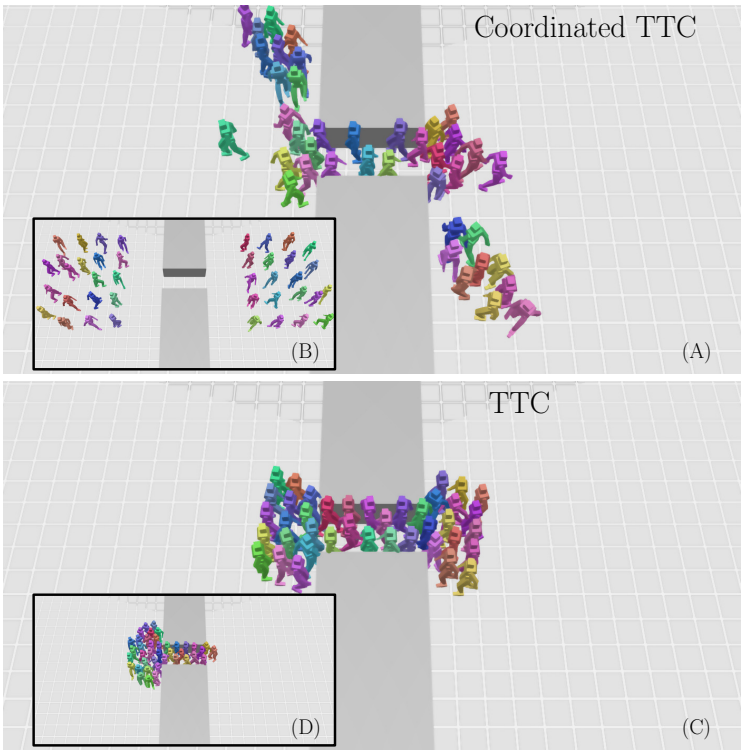

Fig. 1. Simulation of our results for C-TTC (top) compared to a crowd simulation of TTC-Forces [Guy and Karamouzas 2015] (bottom). (A) and (C) compare the two simulations at 7.3 seconds. Note how C-TTC has partitioned the agents into two groups: a queued group to the side of the door, and a laning group passing through the doorway. Inset figure (B) and (D) compare the two simulations at 19.7 seconds when our simulation has finished and while TTC-Forces is in the bottleneck.

congestion. However, a key difficulty in creating such an explicit communication infrastructure is the inherent complexity in designing a protocol for such a system while still maintaining a fast, scalable simulation.

In this paper, we use an optimization-based approach to automatically learn a communication policy between agents that explicitly communicates on channels for local motion planning. The result is the efficient, coordinated motion seen in Figure 1. We refer to this as an "emergent" communication approach because its protocol and channels have no predetermined semantics. Therefore, that system is allowed to learn both *what* is shared between agents and *how* that information is used to affect motion. We propose a unified learning framework that allows agents to simultaneously solve both problems. All of our agents follow the same learned communication policy allowing the development of emergent social norms based on a mutually consistent interpretation of the shared signals (see Figure 2).

Our work has three primary contributions:

- First, we present C-TTC, an algorithm for multi-agent navigation that allows (multi-channel) communication between neighboring agents. We show our method can provide more efficient and coordinated motion than the original TTC-Forces algorithm without communication.

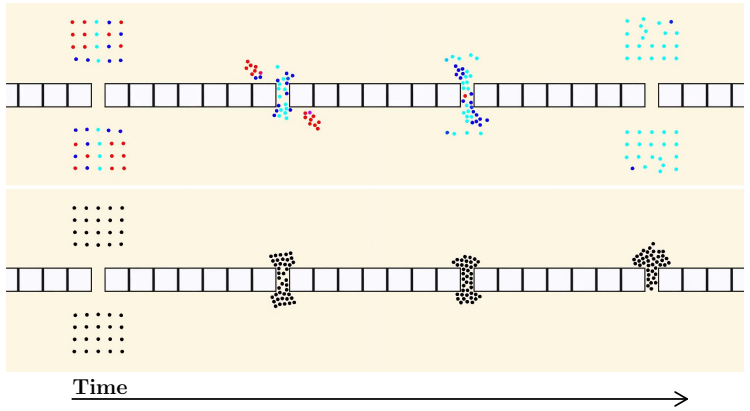

Fig. 2. A communication model (top) optimized and visualized on the Doorway scenario (D) compared to TTC-Forces (bottom). Each vertical frame is at an equivalent time-step. Our method learned to use communication for partitioning the agents into groups where some wait and others follow their lane through the bottleneck. The red agents are communicating that they are waiting, blue agents are pursuing their goal, and cyan agents are preventing crossing into an opposing lane. Once it is efficient to pass, the waiting agents are spatially signaled to pass and then communicate they are pursuing their goal. In contrast, TTC-Forces struggles to form lanes in the dense doorway.

- Second, we show that communication enhanced simulations retain and reinforce the coordination already seen in TTC (e.g., lane formation), while also introducing new types of coordinates behaviors such as grouping up, and waiting for their turn.
- Third, we show the learned communication strategies generalize well, with communication policies trained on one scenario often improving the navigation behavior in new, unrelated scenarios.

The rest of the paper is organized as follows: In Section 2, we review related works involving planning, communication, and learning. We describe in Section 3 our method for coordinated crowd planning by optimizing a communication model. Reported in Section 4 is our implementation and its parameters, including TTC-Forces and PSO for the required collision avoidance and optimizer to use our work. Experimental results demonstrate the tested scenarios, communication, motion efficiency, and scalability of our work in Section 5. A discussion of the limitations and impact of our work is in Section 6.

2 RELATED WORK

There are many works on local multi-agent navigation or collision avoidance, each of which addresses the problem of finding collision-free paths for agents in a scene which have competing routes to their goals. As relevant to this work, these algorithms use a broad variety of strategies for agent dynamics. Reactive methods (e.g., [Karamouzas et al. 2009, 2014; Paris et al. 2007]) rely on forces which repel agents away from collisions, often with analogies to animals [Reynolds 1987], or social structures [Helbing and Molnar 1995]. Geometric methods such as RVO [van den Berg et al. 2008] and ORCA [van den Berg et al. 2011], under minimal assumptions, optimize for a guaranteed set of collision-free paths. Data-driven algorithms attempt to model aspects of human behavior such as their anticipation [Karamouzas et al. 2014], efficiency [Berseth et al. 2014], or patterns of escape [Helbing et al. 2000].

Other multi-agent navigation works have proposed methods of local navigation by augmenting the agents' behaviors with coordination or communication. For example, several authors have used approaches relying on predefined methods of coordination such as forming groups of agents guided by hierarchical rules [Musse and Thalmann 2001], communicating escape routes or roles [Pelechano and Badler 2006], and modeling group interactions [Qiu and Hu 2010]. Using globally coordinated methods is particularly helpful to local navigation for planning around dynamic obstacles [Kapadia et al. 2013], local interactions of variable density [Loscos et al. 2003], or forming dynamic structures [Alonso-Mora et al. 2012]. A universal approach to coordinated group behaviors attempts to composite behaviors onto any preexisting simulation by influencing nearby agents [Schuerman et al. 2010; Yeh et al. 2008] or by influencing nearby agents using a specific method such as velocity obstacles [Kimmel et al. 2012; Ren et al. 2017].

A number of papers have studied the benefits of communication when designed for multi-agent systems. Some works have explored human-designed direct communication with assigned meanings [Balch and Arkin 1994], implicit methods that assume a shared algorithm [Godoy et al. 2018, 2016], or methods of indirect cooperation like stigmergy [Beckers et al. 1994]. Deciding *when* communication is beneficial has been addressed in the literature for non-continuous, infrequent, discretely defined communication policies [Best et al. 2018].

The learning community has used reinforcement learning [Foerster et al. 2016; Ghavamzadeh and Mahadevan 2004; Guestrin et al. 2002; Martinez-Gil et al. 2014; Sukhbaatar et al. 2016; Xuan et al. 2001] and evolutionary techniques [Buzing et al. 2005; Quinn 2001; Wong et al. 2015] to solve more general forms of communicating agents to apply to separate or broader problem domains. Other approaches optimize communication for coordinating control policies between agents [Ghavamzadeh and Mahadevan 2004]. Note that in many of these works, the domains studied (e.g., logic puzzles) do not apply well to navigation as they are neither real-time nor in a continuous space [Foerster et al. 2016; Ghavamzadeh and Mahadevan 2004; Sukhbaatar et al. 2016]. Some papers [Hettiarachchi 2010] also explored the use of evolutionary algorithms to optimize direct communication between agents, often in the pursuit of analyzing the origins of communication [Buzing et al. 2005; Quinn 2001]. Some reinforcement learning approaches have also directly addressed the pedestrian planning problem without communication [Martinez-Gil et al. 2014].

3 APPROACH

We consider the problem of moving agents towards their goals while avoiding collisions with agents and environmental obstacles. Each agent i is circular with a collision radius, r_i, position, $\vec{p}_i$, velocity, $\vec{v}_i$, force, $\vec{F}_i$ and goal position, $\vec{goal}_i$. Obstacles in the environment are modeled as blocks with width and height s. Each agent must have a collision free path which reaches their goal in a time-efficient manner.

Our method, C-TTC, approaches this problem by adding a communication layer that is used to modify the behavior of an underlying collision avoidance model, in this case the TTC-Forces method [Guy and Karamouzas 2015] (See Figure 3). At a high level, C-TTC combines two forces: an avoidance force $\vec{F}_{ai}$ and a coordination force $\vec{F}_{ci}$. $\vec{F}_{ai}$ exists to repel agent i away from imminent collisions but towards their goal, thus guaranteeing that agents eventually reach their goals. $\vec{F}_{ci}$ coordinates the collision avoidance of each agent so as to have more efficient motion (see Algorithm 1). This coordination force is influenced according to some matrix of parameters, $\boldsymbol{M}$. These parameters weight the interaction between agents' observations of each other, their emergent communication, and how they change their motion based on that communication. Therefore, a key aspect of C-TTC is choosing the communication parameters $\boldsymbol{M}$ that produce efficient and coordinated paths.

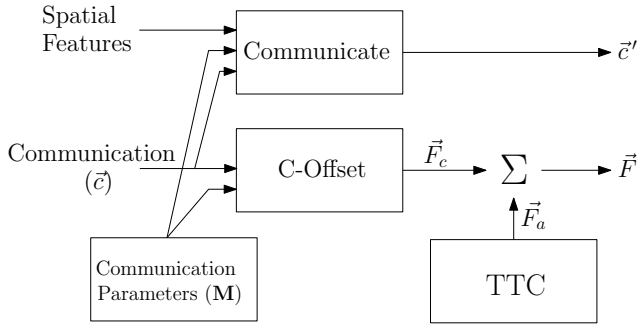

Fig. 3. Flowchart of C-TTC's agent dynamics. We allow the communication features, analogously thought of as "audio", to affect only the forces of the agents, thus emphasizing the communication's effect on coordination. The spatial observations are analogously thought of as "visual" features that can only affect the communication. How these input features affect $\vec{F}_c$ and $\vec{c}'$ is parameterized by $\boldsymbol{M}$, which we optimize for. Everything inherent to TTC, including its input features, is represented within its box.

Different scenarios will have different optimal parameters. Our framework can allow both an "expert" $\boldsymbol{M}$ optimized for an expected environment or a generalized $\boldsymbol{M}$ that supports a wide range of scenarios.

Algorithm 1 Coordinated TTC

Require: $\boldsymbol{M} \leftarrow$ communication matrix
Require: $A \leftarrow$ set of communicating agents
Require: AVOID $\leftarrow$ TTC (force-based collision avoidance)
Require: INTEGRATE $\leftarrow$ Numerical integrator of physics
1: **procedure** $C_{TTC}(\boldsymbol{M}, A)$
2: **for all** $i \in A$ **do**
3: $\vec{F}_{ai} \leftarrow$ AVOID(i, A)
4: $\vec{F}_{ci} \leftarrow$ C-OFFSET$(\boldsymbol{M}, i, A)$ ▷ See Algorithm 2
5: $\vec{F}_i \leftarrow \vec{F}_{ci} + \vec{F}_{ai}$
6: $\vec{p}'_i \leftarrow$ INTEGRATE$(\vec{F}_i)$
7: **for all** $i \in A$ **do**
8: $\vec{p}_i \leftarrow \vec{p}'_i$
9: $\vec{c}_i \leftarrow \vec{c}'_i$

3.1 Decentralized Communication

During every time-step of C-TTC, coordination forces are augmented onto the underlying force-based collision avoidance algorithm (TTC-Forces). The augmentation is done by simply adding the forces (line 5 of C-TTC) of the collision avoidance, $\vec{F}_{ai}$ with our coordination force, $\vec{F}_{ci}$ (Algorithm 2). To compute the coordination force, each agent searches for its nearest neighbor (line 2 of Algorithm 2), computes its coordinate frame $\boldsymbol{T_i}$ (line 3), computes each of the spatial input features with respect to $\boldsymbol{T_i}$ (lines 4), and then computes its coordination force and its own "speech" for the next time-step (line 5). Every time-step the 2D coordinate frame, $\boldsymbol{T}_i$, of the agent is reset by facing it towards the next node in its route and defining the right face of the agent as the cross-product of the facing direction with the canonical up vector. When there are likely no potential interactions

Algorithm 2 Coordination Offset

Require: $\boldsymbol{M} \leftarrow$ communication matrix
Require: $A \leftarrow$ set of communicating agents
Require: $i \leftarrow$ current agent to offset
1: **function** C-OFFSET($\boldsymbol{M}$, i, A)
2: $j \leftarrow \text{NEAREST}(i, A)$
3: Compute $\boldsymbol{T}_i$
4: Compute spatial features $\vec{o}_j^{\,i} = \left[\theta_j^i, s_j^i, \phi_j^i, d_j^i, g_i\right]^T$ w.r.t. $\boldsymbol{T}_i$
5: $\begin{bmatrix} \vec{c}_i' \\ \vec{F}_{ci} \end{bmatrix} \leftarrow \text{CLAMP}\left(\boldsymbol{M} \cdot \begin{bmatrix} \vec{c}_j \\ \vec{o}_j^{\,i} \end{bmatrix}\right)$
6: **if** $\|\vec{goal}_i - \vec{p}_i\| \leq near$ **then**
7: **return** $\vec{0}$
8: **else**
9: **return** $\vec{F}_{ci}$

between agents (i.e. when an agent is near its goal), there is no coordination force as it could only coordinate an agent away from its goal (line 6). However, for the sake of continuity with other agents, agents will continue to communicate even when they have reached the goal (if needed, they are able to communicate this fact with $\boldsymbol{g_i}$). The semantics of the communication channels emerge automatically through the optimization process of optimizing $\boldsymbol{M}$; there is no predetermined meaning to the resulting channels.

For the entire course of a simulation, each agent shares the same parameter set $\boldsymbol{M}$, which is a $(m+2) \times (m+5)$ matrix whose parameters are optimized as described in Section 3.2. There are multiple kinds of input features to $\boldsymbol{M}$, as represented by Figure 4. Each agent holds an m-dimensional vector $\vec{c}_i$ that represents their current projected "audio", as well as $\vec{c}_i'$ which buffers in updates from the model each time-step (line 5 of Algorithm 2 and line 8 of Algorithm 1). This vector always begins the simulation initialized to $\vec{0}$ for every agent. Furthermore, each agent computes input spatial features that it "observes" of j, with respect to i's own coordinate frame, $\boldsymbol{T}_i$.

Each agent receives their input features only from their nearest neighbor, who is referred to as agent j in Algorithm 2. This is because we require a fixed size input out of the variable number of possible inputs from all agents. Furthermore, finding just the nearest neighbor can be efficiently computed with spatial data structures, such as a k-d tree or bounding volume hierarchy, just like with TTC-Forces. Importantly, by reducing the variable number of observable inputs to a fixed size provides our $\boldsymbol{M}$ some invariance to the number of agents in a scenario. Many functions could provide this invariance to the number of agents, such as observing the k nearest neighbors, k "loudest" neighbors where you observe agents with the maximum $\vec{c}_j$ norm weighted by distance, or a distance weighted sum of the neighbors; but, we found the nearest neighbor to be satisfactory in practice. Further, we use only the *one* nearest neighbor, in contrast to a k nearest neighbors to allow for easier optimization of $\boldsymbol{M}$. Using any value of k would be a large number of parameters to optimize for, because the dimensionality of $\boldsymbol{M}$ would grow quadratically: $(mk+2) \times ((m+5)k)$. By using a k of 1 we therefore have a dimensionality of 4×7 with the 2×5 bottom right parameters set to zero.

In combination, we refer to $\vec{c}_j$ and $\vec{o}_j^{\,i}$ in Algorithm 2 as the input features to an agent. The former are the communication features "spoken" by agent j, and the latter are the spatial features of j observed by agent i. Furthermore, $\vec{c}_i'$ refers to agent i's output communication and $\vec{F}_{ci}$ their output coordination force. Referring to equation 1, $\boldsymbol{M_c}$ and $\boldsymbol{M_o}$ are therefore the partitions of $\boldsymbol{M}$ where

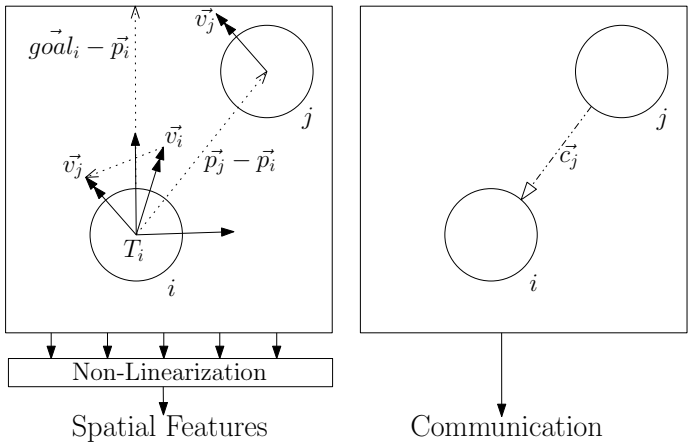

Fig. 4. Representation of the communication and spatial features of Algorithm 2. The audio feature is received from the nearest agent j. Many of the non-linear operations on the diagrammed spatial observations are to make them relative to agent i's coordinate frame $\boldsymbol{T_i}$. More non-linear functions are applied to specific features to simplify the model's training and encourage coordinated behavior.

the communication and observations respectively affect agent i's communication. $\boldsymbol{M_F}$ is then the communication's effect on the motion of the agent i. The resulting $\boldsymbol{M}$ is of the following form:

$$\begin{bmatrix} \vec{c}_i' \\ \vec{F}_{ci} \end{bmatrix} = \textsc{Clamp}\left(\begin{bmatrix} \boldsymbol{M_c} & \boldsymbol{M_o} \\ \boldsymbol{M_F} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \vec{c}_j \\ \vec{o}_j^{\,i} \end{bmatrix} \right). \tag{1}$$

We use zeros for all values in the bottom right of $\boldsymbol{M}$, as can be seen in equation 1, to reduce the number of parameters to optimize. Moreover, this is done to focus on the effect of communication and to avoid being redundant with TTC-Forces. The $\boldsymbol{0}$ partition, were it not zero, would only be affected by the observations of the agent i, and not its communication with some other agent j. Using zeros for the partition forces the agents to communicate in meaningful ways instead of just using their spatial context.

Because $\boldsymbol{M}$ is linear the spatial input features, $\vec{o}_j^{\,i}$ have been carefully chosen and non-linearized to increase the distinctiveness of communication between agents and be analogous to sensory inputs. However, the semantics of $\boldsymbol{M}$ are not predetermined, and the behaviors seen in our results are completely emergent of the optimization. (See Section 5.)

The first k input dimensions of $\boldsymbol{M}$ correspond to agent i listening to its nearest neighbor j's communication values c_j. This is analogous to the sense of hearing in biological agents. The remaining dimensions represent the spatial observations of the environment by agent i, often specifically observing j. Informally, those observations are analogous to the visual and kinesthetic senses in biological agents. A particularly important spatial feature we found to include is the distance of an agent to its goal with some non-linearity. This is so that agents can bias towards behaviors that are independent of their neighbor. Another feature we found important is to have some representation of the relative orientation of the listened-to agent rather than relative displacement.

Finally, a collision avoidance force, $\vec{F}_{ai}$ (from TTC-Forces) is augmented by a force $\vec{F}_{ci}$. We clamp $\vec{F}_{ci}$ to the range [-1, 1] to allow $\vec{F}_{ai}$ to avoid collisions. We also clamp $\vec{c}_i$ to the same range to prevent divergence of $\vec{c}_i$ propagating across agents indefinitely. The non-linearity provided by the clamp helps to give the useful behaviors seen in our results; theoretically, other sigmoidal non-linear functions could be used in the same range. Note that agents can still propagate their $\vec{c}_i$ values;

many of the visualizations found in the Section 5 highlight this when agents switch colors, show a near lack of color, or grow in intensity as they move in some direction.

3.2 Learning

Our evaluation function, shown in Algorithm 3, computes C-TTC's forces every time-step in the form of Algorithm 1. For any given scenario, the following is a summary of how we train an expert model for some scene for use in C-TTC: Many simulations using C-TTC are run to sample and optimize the parameters of $\boldsymbol{M}$ for interaction overhead. Each simulation begins with every agent individually planning a route over a road-map of the environment. During the simulations, anytime an agent cannot see their planned route due to being offset by dynamic interactions they re-plan their route the next time-step. (For performance reasons, a limited number of agents re-plan per simulation time-step and the rest are queued.) Each agent then progresses towards their goals, handling collisions via TTC-Forces as augmented by the communication forces.

The goal of our learning system is to solve the following optimization problem for a given scene S:

$$\boldsymbol{M}_S = \underset{\boldsymbol{M}}{\operatorname{argmin}} f_S(\boldsymbol{M}) \tag{2}$$

where $\boldsymbol{M}_S$ defines an expert communication model for the scene S, although it can be applied to any other scene because Algorithm 2 is agnostic to the scene's initialization and total number of agents.

Definition: Interaction Overhead. This work defines the interaction overhead of agents similarly to [Godoy et al. 2018]. The interaction overhead of the simulation represents the aggregate overhead of each individual agent. The overhead of an agent is the amount of simulated time it took to reach its goal less the time it would take to follow its optimal path:

$$O_i = Time(i) - MinTimeToGoal(i) \tag{3}$$

$$O = E_{i \in A}[O_i] + 3\sigma_{i \in A}(O_i) \tag{4}$$

This captures not only the total time it would take for about 90% of agents to reach their goals (Chebyshev's Inequality), but also gracefully encourages coordinated fairness across agents (fairness being that each gets a similar amount of overhead). This naturally avoids issues with penalizing the last agent taking an outlying length of time, while still penalizing for larger groups arriving much later than the average. This encourages the optimization to minimize the total arrival times of agents and equally weight the amount of overhead each receives.

A gradient free optimizer is used to optimize the communication parameters, $\boldsymbol{M}$, by repeatedly running the simulation given different samples. The training will minimize O for communication and return the optimal O_S and corresponding parameters $\boldsymbol{M}_S$ that produced it.

A global optimizer is required to optimize $\boldsymbol{M}$ because the optimization landscape was found to be noisy, and it needed to quickly explore many drastically different parameters for $\boldsymbol{M}$. Of course, when $\boldsymbol{M}$ is $\boldsymbol{0}$ the communication simulation will perform exactly the same as TTC-Forces. So, our optimization was centered on TTC-Forces by exploring within a window around $\boldsymbol{0}$.

4 IMPLEMENTATION DETAILS

4.1 C-TTC Parameters

Some of the non-linearities we chose for each of the spatial features were specifically to make our method more efficient, particularly using softsign on the goal distance or inverse distance for the relative position. As partially depicted in Figure 4, we defined our spatial features of relative velocity (θ_j^i and s_j^i), relative position (ϕ_j^i and d_j^i), and goal distance (g_i) as follows.

Algorithm 3 Crowd Simulation Evaluation

Require: $\boldsymbol{M} \leftarrow$ communication matrix
Require: $S \leftarrow$ initial scene configuration
function $f_S(\boldsymbol{M})$
  $A \leftarrow S.agents$
  for all $i \in A$ **do**
    PLAN(i)
  $time \leftarrow 0$
  while $time <$ MAXTIME(S) $\wedge$ ($\exists i \in A$ $\neg$DONE(i)) **do**
    C-TTC($\boldsymbol{M}, A$) ▷ See Algorithm 1
    if DONE(i) **then**
      $O_i \leftarrow time -$ MINTIMETOGOAL(i)
    $time \leftarrow time + \delta t$
  for all $i \in A$ **do**
    if $\neg$DONE(i) **then**
      $O_i \leftarrow P\|\vec{goal}_i - \vec{p}_i\| + time -$ MINTIMETOGOAL(i)
  return $O \leftarrow E_{i \in A}[O_i] + 3\sigma_{i \in A}(O_i)$
function DONE(i)
  return $\|\vec{goal}_i - \vec{p}_i\| \leq \epsilon$

Given that the relative velocity of agent j with respect to agent i is $\vec{v}_j^i = \boldsymbol{T}_i \cdot (\vec{v}_j - \vec{v}_i)$ then the relative *speed* between them, which relates the magnitude of correction potentially needed to align two agents, is the following:

$$s_j^i = \|\vec{v}_j^i\| = \|\boldsymbol{T}_i \cdot (\vec{v}_j - \vec{v}_i)\| \tag{5}$$

Then s_j^i is used to compute the *goal-oriented velocity alignment* of the two agents, which corresponds to how much agent j is moving in the way of agent i:

$$\theta_j^i = f_i \cdot (\vec{v}_j - \vec{v}_i)/s_j^i = (\boldsymbol{T}_i \cdot (\vec{v}_j - \vec{v}_i)/s_j^i)_y \tag{6}$$

where $\vec{f}_i$ is the facing of $\boldsymbol{T}_i$ and the canonical y axis of $\boldsymbol{T}_i$'s coordinate frame. This vector can be understood as the vector pointing towards the goal in Figure 4, thus the goal-alignment of this feature.

The *goal-oriented positional alignment*, ϕ_j^i, corresponds to how much agent j is in the way of agent i's path. d_j^i, the *proximity*, emphasizes closer agents (this can be thought as "louder"). Both of these are defined similarly to θ_j^i and s_j^i, respectively:

$$\phi_j^i = f_i \cdot (\vec{p}_j - \vec{p}_i) d_j^i = (\boldsymbol{T}_i \cdot (\vec{p}_j - \vec{p}_i) d_j^i)_y \tag{7}$$

$$d_j^i = \|\vec{p}_j^i\|^{-1} = \|\boldsymbol{T}_i \cdot (\vec{p}_j - \vec{p}_i)\|^{-1} \tag{8}$$

where the relative position of agent j with respect to i is $\vec{p}_j^i = \boldsymbol{T}_i \cdot (\vec{p}_j - \vec{p}_i)$. Note the difference between d_j^i and s_j^i is that d_j^i takes the inverse distance to emphasize closer agents.

The last spatial input feature, g_i, is agent i's distance to its own final goal:

$$g_i = \frac{\|\vec{goal}_i - \vec{p}_i\|}{1 + \|\vec{goal}_i - \vec{p}_i\|} \tag{9}$$

It is non-linearized with softsign to provide a mostly uniform value to $\boldsymbol{M}$ when the agent is far away from its destination, but an increasingly closer to 0 value as it approaches its goal. This uniform value can then be used as a consistent bias, without overpowering the other input features.

For our results, we use a 2-dimensional vector for $\vec{c}$. The penalty in Algorithm 3, P, for agents not finishing within the allowed time we set to 2 because it worked well with our optimizer (larger values discourage divergent behavior more strongly). We fixed the value of δt to 60 Hz time-steps.

4.2 Collision Avoidance: TTC-Forces

Our approach builds off of a baseline collision avoidance algorithm. We assume that it is a force-based algorithm that allows our method to impart an additional coordination force. Here, we use the recent TTC-Forces method [Guy and Karamouzas 2015] because it has shown to have good behavior in practice, and is closely inspired by recent findings of the key PowerLaw relationship that underlies human collision avoidance [Karamouzas et al. 2014].

TTC-Forces completely routes agents with forces to their goals without collisions by predicting the future moment of collision τ. For any agent i, TTC-Forces calculates two combined forces: an avoidance one, $\vec{F}_{ai}$, which repels agent i away from potential collisions as a approximate power law of τ, and a goal one, $\vec{F}_{gi}$, which pulls the agent towards their planned motion. After a time horizon, τ_H, of 5 seconds TTC-forces ignores collision, which balances the assertiveness and conscientiousness of the agents potential collisions [Guy and Karamouzas 2015]. The avoidance force is then the following:

$$\vec{F}_{ai} = \frac{\tau_H - \tau}{\tau} \cdot \frac{\vec{d}}{\|\vec{d}\|} \tag{10}$$

where $\vec{d} = (\vec{p}_j + \vec{v}_j\tau) - (\vec{p}_i + \vec{v}_i\tau)$ is the direction to push agents away from their future collision. The interplay of these forces can be tuned with a constant k that defines the strength of the goal force, which we set to 2 as it balances the two forces well [Guy and Karamouzas 2015]. With that, the goal force can be defined as such:

$$\vec{F}_{gi} = k\left(\vec{v}_{goal} - \vec{v}_i\right) \tag{11}$$

Furthermore, the combined forces are limited to some maximum F_{max}, which we set to 20 because this reasonably balances the stability of the numerical integrator.

To avoid artifacts of symmetric scenes we add a small amount of uniform noise each time-step in the form of a perturbation force. This and any other randomness is seeded with the same value across every training iteration so as to replicate the best iteration for our final renderings.

4.3 Optimization: PSO

Because we use an optimization-based learning approach, choosing a good optimizer is central to achieving good coordination behavior with our framework. Here, we choose the Particle Swarm Optimization (PSO) approach [Zambrano-Bigiarini et al. 2013], as it is a global-optimization method that does not require our objective function to be smooth or differentiable. Additionally, PSO can handle well the presence of multiple local minima which can occur in our training. For similar reasons, PSO has recently been popular in various optimization-based animation work, and has been used directly for optimizing the paths of simulated crowds [Wong et al. 2015].

Briefly, PSO uses a constant number of particles each with a high-dimensional position and velocity that represents a sample of a function that is being optimized. It then operates by randomly exploring the optimization space in directions towards previously found minima. PSO's particles initially randomly sample from -1 to 1 in each dimension of the optimization space. This initial range does not limit the model during the entire optimization; therefore, our optimized models

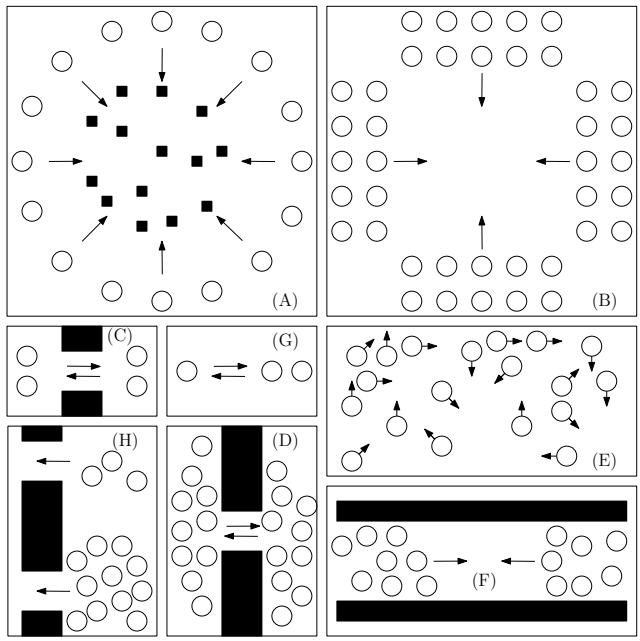

Fig. 5. Illustrations of the test scenes: Circle (A), Intersection (B), Simple Doorway (C), Doorway (D), Crowd (E), Hallway (F), Asymmetric (G), Escape (H). Each scenario is fully detailed in Table 1.

$\boldsymbol{M}$ can contain values larger than 1. The samples are iteratively pulled towards the minima found by each particle to improve an evaluation function, in our method, this is the f of Algorithm 3. Our implementation of PSO uses 40 particles and a maximum of 225 iterations for a total of 9000 function evaluations of f. The evolution of each particle is balanced by an inertial hyper-parameter, w_i, which maintains the motion of the particles, and two more weights, w_l and w_g, which control the gravitation of the particles towards discovered minima. We tuned these hyper-parameters to be $w_i = 0.729$, $w_l = 1.494$, and $w_g = 1.494$ as suggested by the heuristics of [Trelea 2003]. A topology connects the particles with some neighborhood set, which defines the neighborhood's global minimum for w_g. We used a fully connected, global topology.

5 RESULTS

We tested eight scenarios, each of which we chose to cover a broad range of interactions (see 5 for a visual summary of the diversity). For example, we widely varied the number of agents ranging from 3 in Scene G to 200 in Scene E. Some scenes had no obstacles (Scenes B, E, and G) and others had many to produce congestion (Scenes A, C, D, and F). We also varied density of agents with particularly sparse scenarios such as E and G or dense scenarios such as B and F. Furthermore, due to their simplicity, Scenes E, G, and H are all scenes that TTC-Forces already performs reasonably well on, and our method performs similarly. The other scenes, which often contain bottlenecks (Scenes C and D) or dense environments (Scenes A, B, and F), our method performs significantly more efficiently.

As specified in Table 1, we limit the length of every simulation to a specific number of time-steps. This is to prevent the simulation from running indefinitely when an agent attempts a poor coordination strategy that navigates away from its goal. It also allows the optimization to run in a reasonable length of time and not too heavily weight testing runs in which agents get stuck. As a

Table 1. Summaries of each scenario that was trained and tested on. $|A|$ refers to the quantity of agents in the scene. Max time is how long the agents have to reach their goals before terminating the simulation and penalizing their lack of completion.

Name	Scene	$\|A\|$	Max Time	Details
Circle	A	60	40s	60 random .3m posts
Intersection	B	56	40s	Mirrored goals
S. Doorway	C	4	30s	2 on 2
Doorway	D	40	100s	20 per side
Crowd	E	200	30s	Random goals
Hallway	F	40	90s	Initially dense
Asymmetric	G	3	40s	1 on 2
Escape	H	65	60s	one way

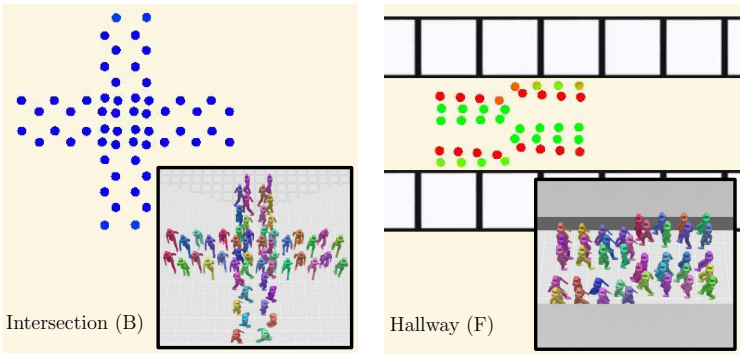

Fig. 6. Highlights of the emergent behavior of C_{TTC} on the Intersection (B) and Hallway (F) scenes. The top subfigure highlights the biasing that agents will learn for coordinated group motion, reminiscent of flocking. Emergent lane behavior is accomplished in the bottom subfigure, ostensibly by agents communicating magnetically; red attracts red and red repels green.

heuristic, these times were chosen to be slightly more than the total time it took for TTC-Forces to finish.

To allow for easy comparison across scenarios, we also define $\hat{O}$ as the overhead of C-TTC normalized to the overhead of TTC-Forces:

$$\hat{O} = \frac{O_{C\text{-}TTC}}{O_{TTC}} \tag{12}$$

A simulation using $\boldsymbol{M}_S$ is rendered and cross-validated for each scenario to evaluate how well this trained model generalizes, and to see the semantic nature of the agents' communication in contrast to the baseline of TTC-Forces. Some of these renderings can be seen in Figure 6 and the resulting $\hat{O}$ overheads from cross-validating can be seen in Figure 7.

For visualizing our communication, we mapped the two values of $\vec{c}_i$ to the CIE-Lab color-space with a brightness of $L = 0.8$. This lets a lack of communication be represented as a desaturated, grey value. Furthermore, each combination of the communication dimensions is represented with a different hue.

See Figure 2 and Figure 6 for highlights of the kinds of emergent coordination behaviors that our method produces. For many of the simpler scenes the resulting communication is relatively

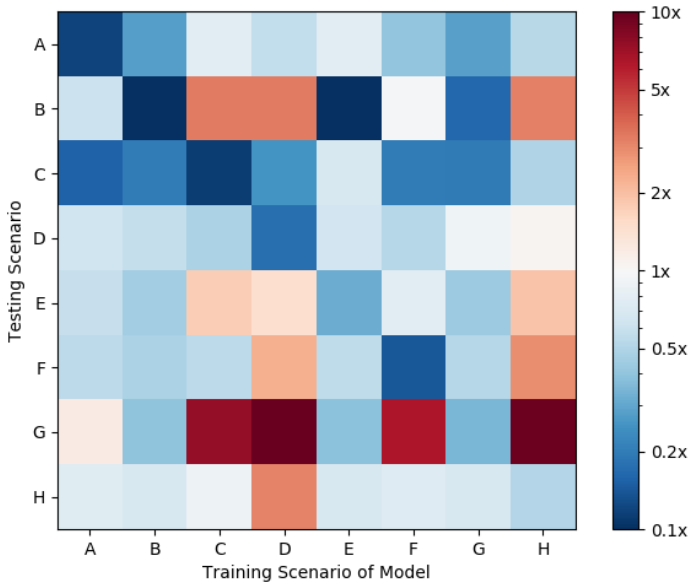

Fig. 7. Cross validation of the trained models to all other scenes. Each cell corresponds to the value of $\hat{O}$ when a model is trained on the scene of its column and tested on the scene of its row. The diagonal is the results of the expert communication model for each scene, which clearly are the best among other models tested on that scene. Blue colors represent overheads less than TTC-Forces's overhead by some multiple according to the colorbar, and red represents overheads greater than TTC-Forces's overhead.

static (Figure 6), the model is able to produce this type of invariant behavior though the proprieties of $\vec{g}_i$ as discussed in Section 4. For scenes with obstacles and dynamic congestion, our agents learn interesting behaviors, such as the following behavior of blue agents after red agents on the Circle (A) scenario (see Video), lane forming behavior in Figure 6, or greeting behavior of the Scene C's model or Scene A's model on Scene C (see Video for latter). As can be seen in the Video, the model for the Escape (H) scene learned to propagate global orientation across the set of agents.

Figure 7 shows every combination of training on one scene and testing on another, producing a matrix of 8×8 $\hat{O}$ values. Overall, our method is clearly more efficient than TTC-Forces in its motion especially in the scenarios it was trained on, as seen by the diagonal of Figure 7. This efficiency leads to the coordination seen in the Video and Figure 2 and Figure 6. Some scenes optimized to a significant improvement on the order of a 90% reduction of TTC-Forces's interaction overhead, such as scenarios A, B, and C.

Some trained communication models generalize better to new scenarios than others. For example the model trained on B generalizes well, with typically less than half as much overhead as TTC-Forces on new scenarios. Additionally, some scenarios are difficult to improve without being explicitly trained on. For example, models tested on G were often worse than TTC-Forces. Typically, models trained on scenes with obstacles learned behaviors "over-fit" to that scenario which result in large overheads on different scenarios (e.g., column D and H).

Values off the diagonal of Figure 7 highlight our method's ability to transfer its model to other tasks it has not encountered. For example, agents trained on the Crowd (E) did just as well on the Intersection (B) as agents that were trained on the Intersection itself, even though they are dissimilar

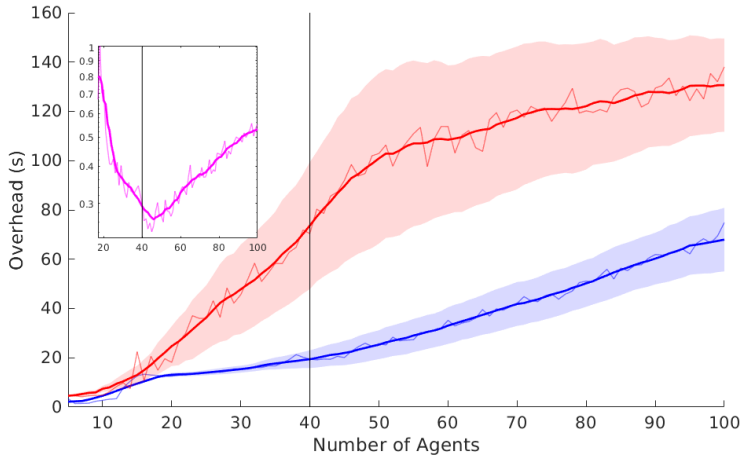

Fig. 8. Scalability of $\boldsymbol{M}$ on Scene D. The outset graph compares TTC in red to C-TTC in blue varied across the number of agents for the Scene D from 5 to 100. Each value of |A| is averaged across 40 seeds of the simulation. A running mean with a window of 11 is drawn over the exact means. The bands around the mean show a running average of ±1 standard deviation. The inset graph represents the normalized $\hat{O}$ for each value of |A|. Note the minimum is around the default number of agents, 40, which is what we optimized $\boldsymbol{M}$ for.

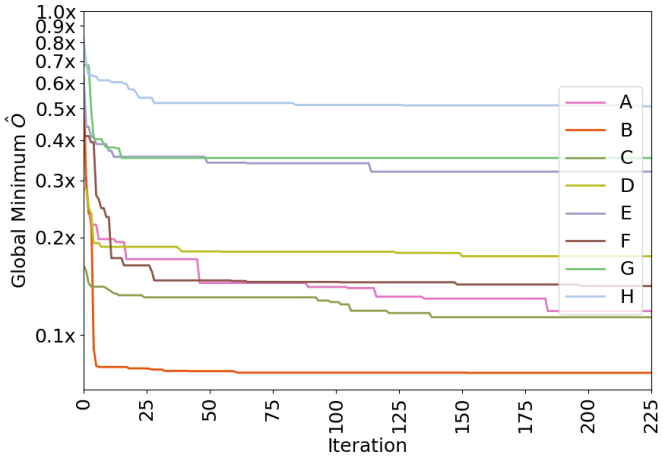

Fig. 9. Plots of the global minimum $\hat{O}$ across all particles at each iteration of training for each scene's model. This validates our training process, especially our evaluation function presented in Algorithm 3.

with different numbers and densities of agents. What our method has learned to communicate in one scenario is generally applicable to collision avoidance in other scenes.

Figure 8 highlights the scalability of any given optimized $\boldsymbol{M}$ across varying number of agents for a given scenario. Note, the $\boldsymbol{M}$ we use for Scene D is optimized for 40 agents yet continues to perform clearly better than TTC well above that, and is as good as TTC beneath approximately 15 agents. Our method also experiences on average one third as much variance under different random seeds of TTC-Force's perturbation force, which indicates its consistently coordinated behavior.

Figure 9 shows the progressive minimization of normalized overhead over each scenario's training. For all models, the optimizer quickly finds a communication strategy that leads to better times than TTC. Note that the early iterations often are better than TTC because we take the best of 40 random particle samples (and then keeping that best for further iterations). Hhowever, further training is clearly required to have more efficient and coordinated behaviors. Most improvement is seen in the first third of iterations, by far most of the improvements are scene in almost every training scenario.

We implemented Algorithm 3 with an efficient program capable of simulating hundreds of agents at thousands of time-steps per second, thus leading to total training times on the order of minutes or hours depending on the scenario. The exact timing of each function evaluation is strongly dependent on our exact implementation, particularly for scenarios with many agents or obstacles. The wall clock time for all 9000 simulations across 225 iterations is roughly 10 hours for the circle scene (*A*), 80 minutes for the intersection scene (*B*), 6 minutes for the simple doorway scene (*C*), 136 minutes for the doorway scene (*D*), 16 hours for the crowd scene (*E*), 89 minutes for the hallway scene (*F*), 3 minutes for the asymmetric scene (*G*), and 245 minutes for the escape scene (*H*). Although all scenarios find significant improvements over TTC-Forces within fractions of those total times.

6 CONCLUSION

In this work we proposed C-TTC, an algorithm for coordinated multi-agent navigation by training agents to use decentralized communication according to some centralized norms. In every scenario we trained for, C-TTC coordinates agents more efficiently to their goals than without communication. Furthermore, the language of communication that the trained models of C-TTC have learned show behavior with global semantics such as those in Figure 6 and in the supplemental Video. We also showed for many of the scenes that the emergent communication policy the agents learned is generally useful to transferred scenarios.

Our results have shown a promising step towards the robust integration of communication with motion planning in crowds. C-TTC agents learn meaningful communication that improves their performance in a dynamic environments in was which are not possible with spatial features alone. There are many other aspects of multi-agent navigation where communication could lead to improved behavior beyond what we address here. For example, learned communication could be applied more globally to coordinate congestion around different exits. Effectively, communication can potentially be used to create a network of agents to coordinate broader goals and actions solely from local interactions.

Limitations. Our method, when transferred to new scenarios, does not always lead to better performance than unmodified TTC-Forces, particularly when applied to scenarios very different than the one it was trained on. Although we only tested TTC-Forces, we could apply our framework to other collision avoidance methods such as Boids [Reynolds 1987] or ORCA [van den Berg et al. 2011]. However, in these cases, it will alter the underlying behavior and may reduce desirable features such as violating the collision-free guarantees in ORCA, or the power law relationship captured in the PowerLaw model [Karamouzas et al. 2014]. PSO as an optimization technique has important limitations when applied to our problem. Even when well tuned, PSO would often struggle to iteratively improve parameter sets $\boldsymbol{M}$ over time. Moreover, PSO is hard to tune due to the time-consuming nature of simulating thousands of crowd simulations.

Future Work. An important avenue for future work is better optimizing the coordination produced by the communication mechanics. While our results show efficient behavior, there are many potential parameters, different input features, and different optimization methods that could further

exploit communication. We are particularly excited about drawing on ideas successful from neural networks. For example, the non-linear pooling of nearby features or using neural networks for each agent's parameters instead of a linear transforms. Performance could perhaps be further improved by providing each agent with memory to dynamically communicate, learn, and plan across a changing network, producing global plans from their local interactions, though this will raise new difficulties with training and generalization. Furthermore, agents could learn to dynamically change their $\boldsymbol{M}$, perhaps by interpolating between separately trained experts, instead of the centralized, static paramater set we use now. We are also excited about the potential of multi-task/multi-objective approaches to optimization, as these provide a natural avenue to train on multiple scenarios at once.

ACKNOWLEDGMENTS

We would like to thank Dr. Maria Gini for her useful discussion and initial feedback on this work and its direction. This work has been supported in part by the NSF through grants #CHS-1526693 and #IIS-1748541.

REFERENCES

Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Roland Siegwart, and Paul Beardsley. 2012. Image and animation display with multiple mobile robots. *The International Journal of Robotics Research* 31, 6 (2012), 753–773.

Tucker Balch and Ronald C Arkin. 1994. Communication in reactive multiagent robotic systems. *Autonomous robots* 1, 1 (1994), 27–52.

Ralph Beckers, OE Holland, and Jean-Louis Deneubourg. 1994. From local actions to global tasks: Stigmergy and collective robotics. In *Artificial life IV*, Vol. 181. 189.

Glen Berseth, Mubbasir Kapadia, Brandon Haworth, and Petros Faloutsos. 2014. SteerFit: Automated parameter fitting for steering algorithms. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 113–122.

Graeme Best, Michael Forrai, Ramgopal R Mettu, and Robert Fitch. 2018. Planning-aware communication for decentralised multi-robot coordination. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1050–1057.

PC Buzing, AE Eiben, and Martijn C Schut. 2005. Emerging communication and cooperation in evolving agent societies. *Journal of Artificial Societies and Social Simulation* 8, 1 (2005).

Jakob Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*. 2137–2145.

Mohammad Ghavamzadeh and Sridhar Mahadevan. 2004. Learning to communicate and act using hierarchical reinforcement learning. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*. IEEE Computer Society, 1114–1121.

Julio Godoy, Tiannan Chen, Stephen J Guy, Ioannis Karamouzas, and Maria Gini. 2018. ALAN: adaptive learning for multi-agent navigation. *Autonomous Robots* (2018), 1–20.

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

Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. 2002. Coordinated reinforcement learning. In *ICML*, Vol. 2. Citeseer, 227–234.

Stephen J Guy and Ioannis Karamouzas. 2015. Guide to Anticipatory Collision Avoidance. In *Game AI Pro 2*, Steve Rabin (Ed.). CRC Press, Chapter 19, 195–208.

Dirk Helbing, Illés Farkas, and Tamas Vicsek. 2000. Simulating dynamical features of escape panic. *Nature* 407, 6803 (2000), 487.

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

Suranga Hettiarachchi. 2010. An evolutionary approach to swarm adaptation in dense environments. In *ICCAS 2010*. IEEE, 962–966.

Mubbasir Kapadia, Alejandro Beacco, Francisco Garcia, Vivek Reddy, Nuria Pelechano, and Norman I Badler. 2013. Multi-domain real-time planning in dynamic environments. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics symposium on computer animation*. ACM, 115–124.

Ioannis Karamouzas, Peter Heil, Pascal Van Beek, and Mark H Overmars. 2009. A predictive collision avoidance model for pedestrian simulation. In *International Workshop on Motion in Games*. Springer, 41–52.

Ioannis Karamouzas, Brian Skinner, and Stephen J Guy. 2014. Universal power law governing pedestrian interactions. *Physical review letters* 113, 23 (2014), 238701.

Andrew Kimmel, Andrew Dobson, and Kostas Bekris. 2012. Maintaining team coherence under the velocity obstacle framework. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 247–256.

Celine Loscos, David Marchal, and Alexandre Meyer. 2003. Intuitive crowd behavior in dense urban environments using local laws. In *Proceedings of Theory and Practice of Computer Graphics, 2003.* IEEE, 122–129.

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

Soraia Raupp Musse and Daniel Thalmann. 2001. Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics* 7, 2 (2001), 152–164.

Sébastien Paris, Julien Pettré, and Stéphane Donikian. 2007. Pedestrian Reactive Navigation for Crowd Simulation: a Predictive Approach. *Comput. Graph. Forum* 26 (2007), 665–674.

Nuria Pelechano and Norman I Badler. 2006. Modeling crowd and trained leader behavior during building evacuation. *IEEE computer graphics and applications* 26, 6 (2006), 80–86.

Fasheng Qiu and Xiaolin Hu. 2010. Modeling group structures in pedestrian crowd simulation. *Simulation Modelling Practice and Theory* 18, 2 (2010), 190–205.

Matt Quinn. 2001. Evolving communication without dedicated communication channels. In *European Conference on Artificial Life*. Springer, 357–366.

Zhiguo Ren, Panayiotis Charalambous, Julien Bruneau, Qunsheng Peng, and Julien Pettré. 2017. Group Modeling: A Unified Velocity-Based Approach. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 45–56.

Craig W Reynolds. 1987. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH computer graphics* 21, 4 (1987), 25–34.

Matthew Schuerman, Shawn Singh, Mubbasir Kapadia, and Petros Faloutsos. 2010. Situation agents: agent-based externalized steering logic. *Computer Animation and Virtual Worlds* 21, 3-4 (2010), 267–276.

Sainbayar Sukhbaatar, Rob Fergus, et al. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*. 2244–2252.

Ioan Cristian Trelea. 2003. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters* 85, 6 (2003), 317–325.

Jur van den Berg, Stephen Guy, Ming Lin, and Dinesh Manocha. 2011. Reciprocal n-body collision avoidance. *Robotics research* (2011), 3–19.

Jur van den Berg, Ming Lin, and Dinesh Manocha. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*. IEEE, 1928–1935.

Sai-Keung Wong, Pao-Kun Tang, Fu-Shun Li, Zong-Min Wang, and Shih-Ting Yu. 2015. Guidance path scheduling using particle swarm optimization in crowd simulation. *Computer Animation and Virtual Worlds* 26, 3-4 (2015), 387–395.

Ping Xuan, Victor Lesser, and Shlomo Zilberstein. 2001. Communication decisions in multi-agent cooperation: Model and experiments. In *Proceedings of the fifth international conference on Autonomous agents*. ACM, 616–623.

Hengchin Yeh, Sean Curtis, Sachin Patil, Jur van den Berg, Dinesh Manocha, and Ming Lin. 2008. Composite agents. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 39–47.

Mauricio Zambrano-Bigiarini, Maurice Clerc, and Rodrigo Rojas. 2013. Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements. In *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2337–2344.