LEARNING TO COMMUNICATE FOR COORDINATED MULTI-AGENT NAVIGATION

A THESIS

SUBMITTED TO THE FACULTY OF THE

UNIVERSITY OF MINNESOTA

BY

Dalton James Hildreth

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

WITH ADVISING FROM

Dr. Stephen J. Guy

JUNE 2019

ACKNOWLEDGEMENTS

First, I would like to acknowledge the phenomenal amount of guidance, teaching, and advice I have received from my supervisor: Dr. Stephen J. Guy. I know I am moving on, but you have had a formative impact on my life and career that I will never forget. I don't think I have said it enough: genuinely, I thank you for all the time you have given me during these past three years.

My thanks go to my thesis committee, Dr. Maria Gini, Dr. Andrew Lamperski, and again Dr. Stephen J. Guy, for reviewing my thesis and taking the time to give me feedback on this work. I especially want to acknowledge Dr. Gini again for her useful discussion and initial feedback on this work and its direction when it was originally devised for her seminar class.

I give my gratitude to all of my family, friends, and members of Dr. Guy's lab for their support. Specifically, I am grateful for Bobby Davis, Nick Sohre, and Tiannan Chen for the many times they have given me critical feedback on my work, helpful conversations about ideas, or programming aid.

I acknowledge that portions of this manuscript are published in ACM's PACMCGIT journal, but this is a permissible, significant addition as per the original license agreement. This work has been supported in part by the NSF through grants #CHS-1526693 and #IIS-1748541.

To my mother, who taught me how to learn:
“stop, breathe, and simplify".
Mom, I feel fortunate for this past you have gifted
to me and the future you have nurtured in me
where I can say I am a scientist.

ABSTRACT

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 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".

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 Premise

Capturing the efficient, coordinated behavior common in human motion is applicable to a number of fields such as architecture, video games, movies, and virtual reality. For example, architects will design through an iterative process of simulation and evaluation of a building's traffic flow to accommodate dense crowds [23, 44]. Moreover, computer games often must simulate under real-time constraints scores of characters moving in coordinated formations or realistic crowds [46]. And, for other similar applications, 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. Furthermore, crowd simulations optimized for navigational efficiency provide behavior akin to realistic, human motion [7, 19], and so this same efficiency can produce cooperative, coordinated kinds of motion. It is a natural, linguistic pattern of human behavior to have idioms, norms, and customs which allow us to efficiently coordinate large formations from locally communicated interactions. So, the central motivation of this work is addressing how a multi-agent system could learn norms to explicitly communicate for the purpose of coordinated behaviors.

The process of motion planning is often split in two: a global stage of planning a route to an agent's intended destination, and a local stage of navigating around the static or dynamic obstacles that invade the agent's anticipated trajectory. Because this work is concerned with coordinated behavior, the latter problem of navigation is the most relevant as it is what can lead from local interactions to global formations. However, because of this decentralization there will be dynamic agents that would cause anticipated collisions. Models do exist to guarantee, if they exist, collision free trajectories under real-time (e.g., ORCA [6]); though, many local minima can occur, where one

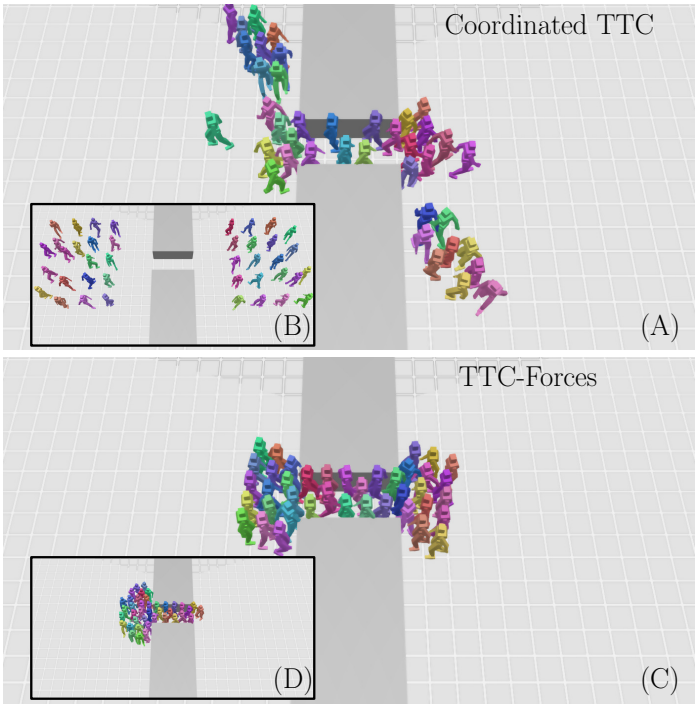


FIGURE 1.1: Simulation of our results for C-TTC (top) compared to a crowd simulation of TTC-Forces [18] (bottom). (A) and (C) compare the two simulations at 7.3 seconds. Note how C-TTC has partitioned the agents into two groups: a queued group to the side of the door, and a laning group passing through 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.

agent can disadvantage others to try optimizing its own navigation. In complex, dynamic scenes, the global efficiency of agents is therefore not guaranteed, as unnatural behavior, congested traffic flow, and even deadlocks can occur from the independently optimal navigation. One solution for global efficiency is to centrally plan, as if one authoritative agent decided every agents' trajectory. However, execution must be in real-time to be scalable for the frequent re-navigation of many agents, or a decentralized system may be required by limited sensory ranges So, instead, navigation ought to continuously, locally coordinate (i.e. communicate). Thus, some policy of communication, which in this work is a non-grammatical language that defines the motion dynamics, must be used to produce coordination for efficient behavior. Using this kind of communication, the local observations of agents can coordinate into globally efficient motion (for example, Figure 1.1).

While there has been much recent work on how simulated agents can make independent decisions in the process of navigation, for example using techniques such as ORCA [6], not as much is known about how agents can *learn* to communicate with each other in ways that improve the efficiency of navigation. Agents who are able to “talk" to their nearby neighbors should, in theory, be able to propogate critical information needed to better coordinate their motion and avoid sources of 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.

Therefore, the thesis of this research is how we can jointly model learning to communicate with (and for) coordinated navigation. To have decentralized, real-time, globally efficient, and coordinated navigation is elegantly resolved with a learned norm of communication. The problem of learning to effectively communicate is thus transformed into producing efficient, coordinated navigation of a simulated crowd. To emulate such norms the policy must not be human designed; this fits navigation because it is impractical to account for all of the myriad of possible local minima that can occur.

1.2 Contributions

To learn to communicate, we use an optimization-based approach to specify a communication policy between agents that controls the explicit communication between agents over multiple channels for local motion planning. The result is the efficient, coordinated motion seen in Figure 1.1. 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 1.2).

This work has three primary contributions to the joint problems of learning communication and multi-agent navigation:

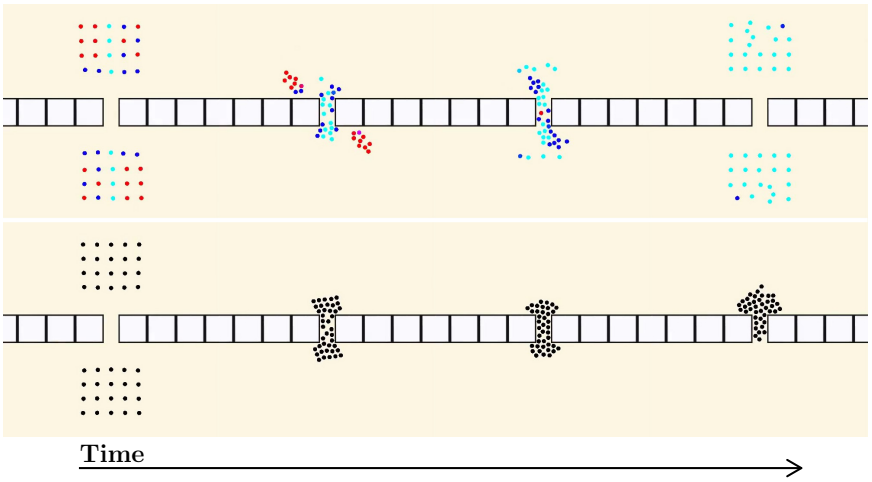


FIGURE 1.2: A communication model (top) optimized and visualized on the Doorway scenario (*D*) compared to TTC-Forces (bottom). Each vertical frame is at an equivalent time-step. Our method learned to use communication for partitioning the agents into groups where some wait and others follow their lane 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.

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

- Second, we show that simulations enhanced with communication retain and reinforce the coordination already seen in TTC-Forces (such as lane formation), while introducing new coordinated behaviors such as partitioning into groups, propagated motion, and agents waiting for their turn.

- Third, we show the learned communication strategies generalize well; communication policies trained on one scenario often improve the navigation behavior in new, unrelated scenarios. Moreover, we analyze how this generality highlights the linguistic, centralized norms of

C-TTC by varying its components.

Overview. The rest of this thesis is organized according to the following summary: Chapter 2 reviews related work involving multi-agent navigation, decentralized multi-agent systems using communication, and how learning has previously been used for multi-agent tasks. Chapter 3 describes the details of C-TTC, an optimization-based method of communication for coordinated motion planning. Reported in Chapter 4 is the tested implementation of C-TTC and its parametric components. This includes a description of TTC-Forces and PSO, which are respectively required for the baseline collision avoidance and optimizer of C-TTC. We analyze our work in Chapter 5 with a set of scenarios applied over numerous experiments testing C-TTC's behavior, efficiency of motion, and scalability. Furthermore, Chapter 5 analyzes how the various components of C-TTC affect its linguistic properties, motion, and generalized efficiency. Chapter 6 discusses the limitations and impact of this work, especially as it relates to possible extensions and further questions.

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., [26, 35]) rely on local forces which repel agents away from collisions. These methods often use analogies to animals, such as with the seminal work of boids [41] or social structures [21]. Geometric methods such as RVO [5] and ORCA [6], 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 [25], efficiency [7], or patterns of escape [20].

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 [33], communicating escape routes or roles [37], and modeling group interactions [3, 38]. Using globally coordinated methods is particularly helpful to local navigation for planning around dynamic obstacles [24], local interactions of variable density [29], or forming dynamic structures [1]. A universal approach to coordinated group behaviors attempts to composite behaviors onto any preexisting simulation by influencing nearby agents [42, 49] or by influencing nearby agents using a specific method such as velocity obstacles [28, 40].

A number of papers have studied the benefits of communication and coordination when designed for multi-agent systems. Some works have explored human-designed direct communication with assigned meanings [2], implicit methods that assume a shared algorithm [15, 16], or methods

of indirect cooperation like stigmergy [4, 27]. Often communication is used in decentralized multi-agent systems to solve planning problems [12, 34, 36] Deciding *when* communication is beneficial has been addressed in the literature for non-continuous, infrequent, discretely defined communication policies [8]. One work addresses how communication and coordination can be used to learn a policy instead of using it for the policy's task [17].

The learning community has used reinforcement learning [11, 13, 17, 30, 31, 43, 48] and evolutionary techniques [9, 39, 47] 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 [13, 31]. 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 [11, 13, 43]. Some papers [22] also explored the use of evolutionary algorithms to optimize direct communication between agents, often in the pursuit of analyzing the origins of communication [9, 39]. Some reinforcement learning approaches have also directly addressed the pedestrian planning problem, although without communication [30].

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 [18] (See Figure 3.1). 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. 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.

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

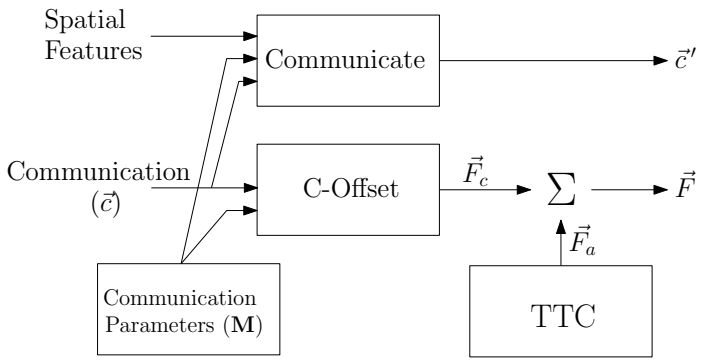


FIGURE 3.1: Flowchart of C-TTC's agent dynamics. We allow the communication features, analogously thought of as "audio", to affect only the forces of the agents, thus emphasizing the communication's effect on coordination. The spatial observations are analogously thought of as "visual" features that 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.

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

procedure $\text{C}_{TTC}(\boldsymbol{M}, A)$
  for all $i \in A$ **do**
    $\vec{F}_{ai} \leftarrow \text{AVOID}(i, A)$
    $\vec{F}_{ci} \leftarrow \text{C-OFFSET}(\boldsymbol{M}, i, A)$ ▷ See Algorithm 2
    $\vec{F}_i \leftarrow \vec{F}_{ci} + \vec{F}_{ai}$
    $\vec{p}'_i \leftarrow \text{INTEGRATE}(\vec{F}_i)$
  for all $i \in A$ **do**
    $\vec{p}_i \leftarrow \vec{p}'_i$
    $\vec{c}_i \leftarrow \vec{c}'_i$

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** $\vec{F}_{ci}$

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 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 3.2. 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

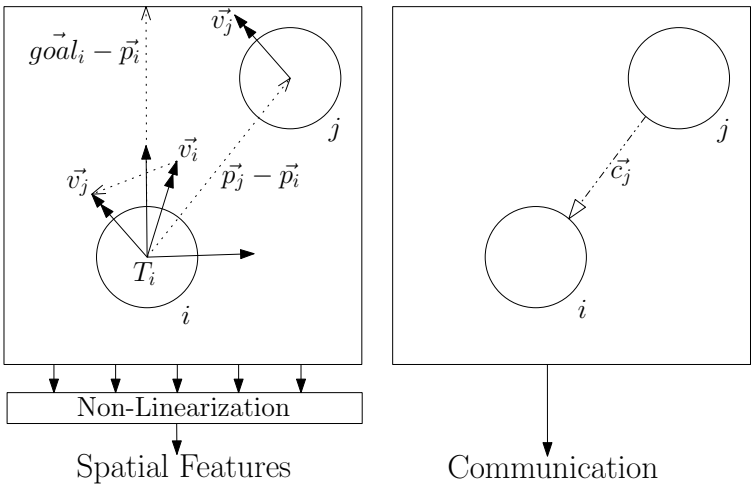


FIGURE 3.2: Representation of the communication and spatial features of Algorithm 2. The audio feature is received from the nearest agent j. Many of the non-linear operations on the diagrammed spatial observations are to make them relative to agent i's coordinate frame $\boldsymbol{T}_i$. More non-linear functions are applied to specific features to simplify the model's training and encourage coordinated behavior.

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 with k:

$$(mk+2)\times((m+5)k) \tag{3.1}$$

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 3.2, $\boldsymbol{M}_c$ and $\boldsymbol{M}_o$ are therefore the partitions of $\boldsymbol{M}$ where 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:

$$\begin{bmatrix}\vec{c}_i' \\ \vec{F}_{ci}\end{bmatrix} = \textsc{Clamp}\left(\begin{bmatrix}\boldsymbol{M}_c & \boldsymbol{M}_o \\ \boldsymbol{M}_F & \mathbf{0}\end{bmatrix}\begin{bmatrix}\vec{c}_j \\ \vec{o}_j^i\end{bmatrix}\right) \tag{3.2}$$

We use zeros for all values in the bottom right of $\boldsymbol{M}$, as can be seen in equation 3.2, 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 $\mathbf{0}$ 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 Chapter 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 Chapter 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:

$$\boldsymbol{M}_S = \underset{\boldsymbol{M}}{\operatorname{argmin}} f_S(\boldsymbol{M}) \tag{3.3}$$

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.

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 $time < \text{MAXTIME}(S) \wedge (\exists i \in A\ \neg\text{DONE}(i))$ **do**
    C-TTC($\boldsymbol{M}$,A) ▷ See Algorithm 1
    if DONE(i) **then**
      $O_i \leftarrow time - \text{MINTIMETOGOAL}(i)$
    $time \leftarrow time + \delta t$
  for all $i \in A$ **do**
    if $\neg$DONE(i) **then**
      $O_i \leftarrow P\|\vec{goal}_i - \vec{p}_i\| + time - \text{MINTIMETOGOAL}(i)$
  return $O \leftarrow E_{i\in A}[O_i] + 3\sigma_{i\in A}(O_i)$
function DONE(i)
  return $\|\vec{goal}_i - \vec{p}_i\| \leq \varepsilon$

Definition: Interaction Overhead This work defines the interaction overhead of agents similarly to [15]. 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:

$$O_i = Time(i) - MinTimeToGoal(i) \tag{3.4}$$

$$O = E_{i\in A}[O_i] + 3\sigma_{i\in A}(O_i) \tag{3.5}$$

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 $\mathbf{0}$ 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 $\mathbf{0}$.

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 3.2, 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.

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:

$$s_j^i = \|\vec{v}_j^i\| = \|\boldsymbol{T}_i \cdot (\vec{v}_j - \vec{v}_i)\| \tag{4.1}$$

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:

$$\theta_j^i = f_i \cdot (\vec{v}_j - \vec{v}_i)/s_j^i = (\boldsymbol{T}_i \cdot (\vec{v}_j - \vec{v}_i)/s_j^i)_y \tag{4.2}$$

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 3.2, 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:

$$\phi_j^i = f_i \cdot (\vec{p}_j - \vec{p}_i) d_j^i = (\boldsymbol{T}_i \cdot (\vec{p}_j - \vec{p}_i) d_j^i)_y \tag{4.3}$$

$$d_j^i = \|\vec{p}_j^i\|^{-1} = \|\boldsymbol{T}_i \cdot (\vec{p}_j - \vec{p}_i)\|^{-1} \tag{4.4}$$

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.

Note that both of ϕ_j^i and θ_j^i were tested using 2 dimensions instead of taking only the forward, y dimension. Experimentally we found only considering the y axis to perform better, however. This may be because of equivalent optimization times for both methods, which could have led to worse overheads when using both dimensions because it would have more parameters.

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

$$g_i = \frac{\|\vec{goal}_i - \vec{p}_i\|}{1 + \|\vec{goal}_i - \vec{p}_i\|} \tag{4.5}$$

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 [18] 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 [25].

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 [18]. The avoidance force is then the following:

$$\vec{F}_{ai} = \frac{\tau_H - \tau}{\tau} \cdot \frac{\vec{d}}{\|\vec{d}\|} \tag{4.6}$$

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 [18]. With that, the goal force can be defined as such:

$$\vec{F}_{gi} = k\left(\vec{v}_{goal} - \vec{v}_i\right) \tag{4.7}$$

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 [50], 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 [47].

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 $\boldsymbol{M}$ 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 [45]. 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 EXPERIMENTAL ANALYSIS

5.1 Experiments

We tested eight scenarios, each of which we chose to cover a broad range of interactions (see 5.1 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 B, C, and G are all scenes that TTC-Forces already has little overhead (less than 5 seconds) for, and our method performs similarly. The other scenes, which often contain bottlenecks (Scenes D and H) or dense environments (Scenes A, E, and F), our method performs significantly more efficiently.

As specified in Table 5.1, 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 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:

$$\hat{O} = \frac{O_{C\text{-}TTC}}{O_{TTC}} \tag{5.1}$$

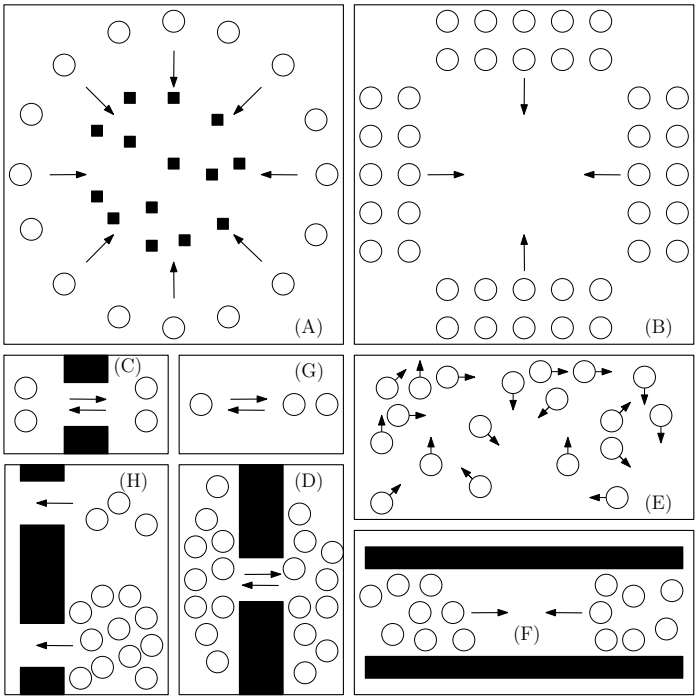


FIGURE 5.1: Illustrations of the test scenes: Circle (A), Intersection (B), Simple Doorway (C), Doorway (D), Crowd (E), Hallway (F), Asymmetric (G), Escape (H). Each scenario is fully detailed in Table 5.1.

TABLE 5.1: 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

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.

5.2 Emergent Behavior

A simulation using $\boldsymbol{M}_S$ is rendered and cross-validated for each scenario to evaluate how well a trained “expert" 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 5.2 and the resulting $\hat{O}$ overheads from cross-validating can be seen in Figure 5.3. In addition to Figure 5.2, the supplemental Video highlights how the constructs presented in Section 3.2 when given the metrics of Section 3.1 generate these kinds of behaviors: agents spontaneously greeting, leading others, or politely letting others forward to increase traffic flow. Scenario H of Figure 5.2, in particular, illustrates the local-to-global propagating capabilities of communication. In this instance, the blue agents have just finished a series of propagated communication, which in the following seconds will lower the density of the exiting side (as blue agents in the Escape (H) scenario are biased to move to the right).

See Figure 1.2 and Figure 5.2 for highlights of the kinds of emergent coordination behaviors that our method produces. For many of the simpler scenes the resulting communication is relatively static (Scene *B*of Figure 5.2), the model is able to produce this type of invariant behavior though the proprieties of $\vec{g}_i$ as discussed in Chapter 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 5.2, or greeting behavior of Scene C’s expert or Scene A’s expert on Scene C (see Video for latter).

5.3 Generalization

Figure 5.3 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 5.3. This efficiency leads to the coordination seen in the Video and Figure 1.2 and Figure 5.2. Some scenes

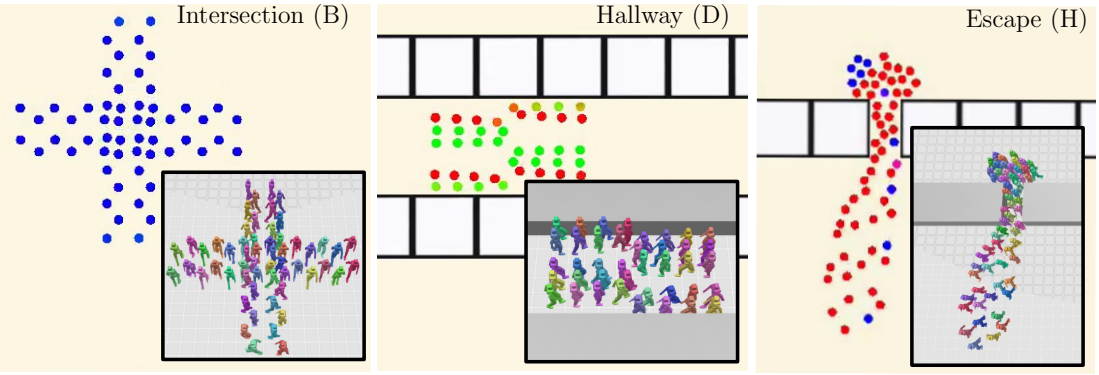


FIGURE 5.2: Highlights of the emergent behavior of C_{TTC} on the Intersection (B), Hallway (F), and Escape (H) scenes. The left sub-figure highlights the biasing that agents will learn for coordinated group motion, reminiscent of flocking. Emergent lane behavior is accomplished in the middle sub-figure, seemingly by agents coordinating into similar hue and direction clumps. In the right sub-figure agents have learned to propagate “blue" values across the doorway; this allows for efficient densities on both sides because of blue agents being biased to move to the right.

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 5.3 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 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 5.4 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

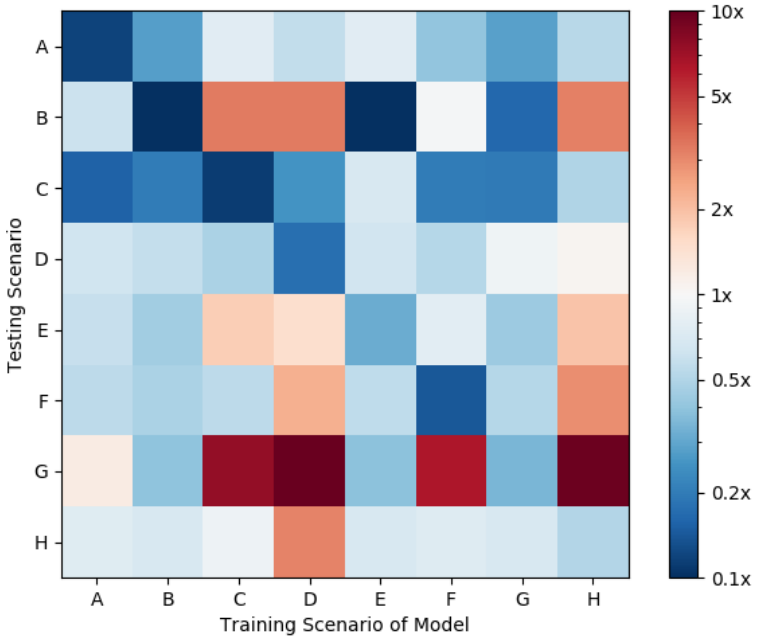


FIGURE 5.3: Cross validation of the trained models to all other scenes. Each cell corresponds to the value of $\hat{O}$ when a model is trained on the scene of its column and tested on the scene of its row. The diagonal is the results of the expert communication model for each scene, which clearly are 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.

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.

5.4 Optimization

Figure 5.5 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

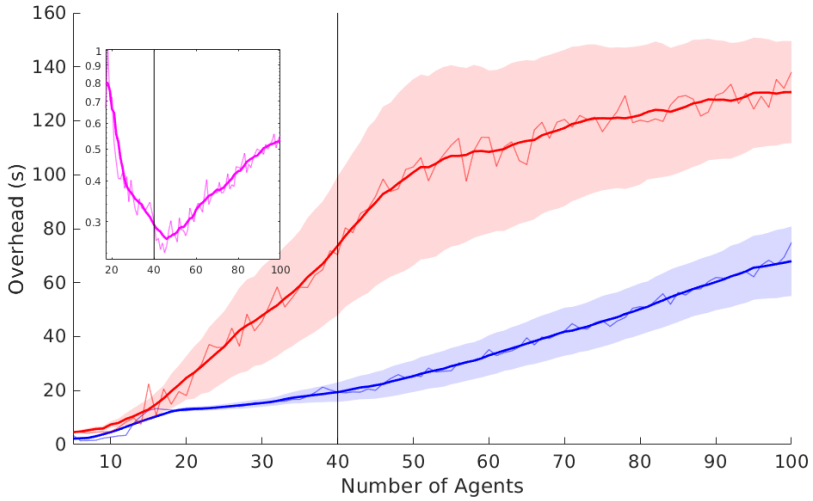


FIGURE 5.4: Scalability of $\boldsymbol{M}$ on Scene D. The outset graph compares TTC in red to C-TTC in blue varied across the number of agents for the Scene D from 5 to 100. Each value of |A| is averaged across 40 seeds of the simulation. A running mean with a window of 11 is drawn over the exact means. 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.

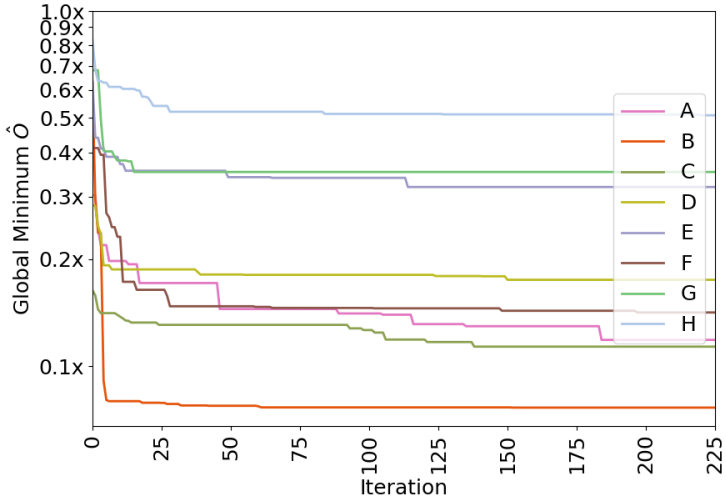


FIGURE 5.5: Plots of the global minimum $\hat{O}$ across all particles at each iteration of training for each scene's model. This validates our training process, especially our evaluation function presented in Algorithm 3.

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.

5.4.1 Multi-Scene Optimization

It is difficult to train on multiple scenes, but it does work to use simple loss functions such as summation. However, these multi-task optimization solutions in our experiments were not clearly better than a well-chosen expert. Many of the difficulties related to training on multiple scenes comes from there existing two kinds of behavior: uniform and dynamic. These will develop depending on the scenes included in an optimization and the loss function used.

Simple scenes either have “short" TTC overheads (roughly less than 5 seconds) or largely straight optimal paths. Complex scenes either have “long" TTC overheads or more complex optimal paths that are likely to lead to congestion. When optimized and tested on, simple scenes tend to produce more uniform behavior as seen in Figure 5.2 for Scene B. By contrast, complex scenes tend to produce more dynamic behavior such as that of Figure 1.2.

There is asymmetry in Fig 5.3 due to the varying difficulty of transferring from or to different scenes. Some scenes such as the Crowd tend to generalize to other scenes better as they tend to have a bias in their motion that is more independent of observational change, thus making them somewhat invariant to their environment. In contrast, some scenes such as the Doorway (D) tend to

not be generalized as well *to* (such as by the Assymetric scene (G)), and tend to be the only scene that does relatively well on itself because they require complex observations to be very efficient. Furthermore, these difficult scenarios tend to not generalize as well, either, due to their strong dependency on observational cues.

The difficulty of multi-scene optimization is rooted in the varying complexities of the scenarios, especially with regards to whether their optimal $\boldsymbol{M}$ would produce uniformly biased or dynamic, context-dependent behavior. This difficulty is also reasonably associated with the simple and complex scenes. Such that simple scenes tend to be good to transfer from, and complex scenes are hard to transfer from. Simple scenes also tend to be hard to transfer to, whereas complex scenes are easy to transfer *something* better to, but not to the same order of magnitude as if it were trained for it. Effectively, the difficulty is a regularizing trade-off between parameters which uniformly bias their agents into generally good, but not relatively the best, behaviors, and parameters which influence the agents' behavior to exploit their complex scenario the best, but not other scenarios very well.

Incorporating simple scenes with complex scenes in a single multi-scene optimization tends to regularize both into a uniform behavior or both into a dynamic behavior depending on the loss function:

- This loss function tends to produce dynamic behavior because it penalizes long overheads relatively more, so it favors minimizing scenarios with long overheads:

$$\frac{\sum O}{\sum O_{TTC}} \tag{5.2}$$

- In contrast, this loss function gives more uniform behavior in our experiments. This is because it penalizes short overhead scenarios relatively more, so it will favor minimizing short scenarios:

$$\frac{\sum \hat{O}}{\sum \hat{O_{TTC}} = |S|} \tag{5.3}$$

To properly merge the above two loss functions would require an internal neural network, memory units, Pareto-set optimization, or a controlled mixing of expert $\boldsymbol{M}$'s.

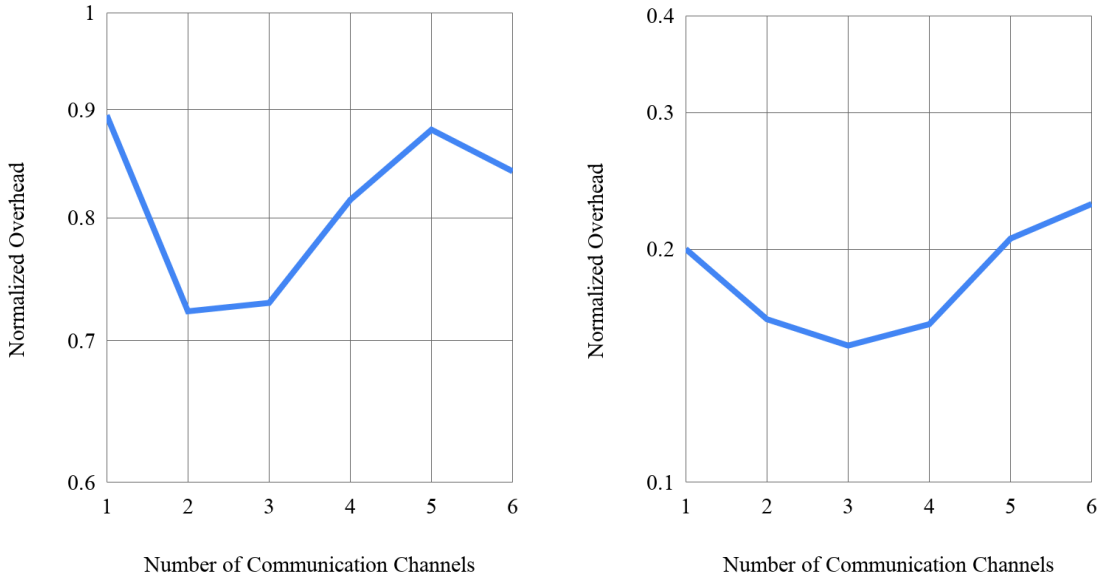
5.5 Communication Channels

Were it not for using the same amount of function evaluations for optimizating an $\boldsymbol{M}$ in all of our experiments, using a larger number m of communication channels should only improve as any unnecessary channels could be zeroed out. Using larger values of m results in larger numbers of parameters to optimize for, similarly to k from Section 3.1, the dimensionality would grow quadratically with m (see Equation 3.1). Therefore, we would expect some number of channels for which C-TTC can be meta-optimized (i.e. the typical overheads of any trained model are minimized).

Figure 5.6 aggregates the overheads for a matrix of cross-validations (just like in Figure 5.3) as the number of communication channels is varied, thus showcasing when m minimizes the overhead. Using a median over the means of each test scenario, instead of a mean over the whole matrix of cross-validations, provides robustness to scenarios of outlying difficulty (for example, Scenario *G*, as seen in the test row of Figure 5.3, would skew the aggregate to seem worse than C-TTC's typical overheads). The numeric impact on overhead is minor, with the exact effects depending on which scenario was trained for. Furthermore, confidence in these results would require significantly more samples because the variance between scenarios is larger than the differences between the means of the different trials. For the scenarios specifically trained for (Figure 5.6b), $m = 3$ would appear quantitatively better. However, when evaluating the generalizability of varying channel numbers (see Figure 5.6a), $m = 2$ is seemingly equivalent to $m = 3$, implying again the variance in these results. Even when generalized, though, the median of the per-test means all perform better than TTC-Forces.

Figure 5.7 highlights a consistent difference between varying channel amounts, a pattern consistent across all other tested variations of C-TTC: $m = 3$ forms waiting lines more naturally than $m = 2$ on Scene *D*. Across all scenarios, the clearest differences between varying numbers of channels is visually, which implies some amount of control over what emerges from C-TTC.

Using $m = 3$ is probably better than $m = 2$ based on the visual results of Figure 5.7 and nearly equivalent results of Figure 5.6, although both are unequivocally better than TTC-Forces. By varying the number of communication channels, Figure 5.7 highlights the intuition that using more channels inherently can allow for more efficient, coordinated, and natural behavior.



(A) Median of per-test means.

(B) Median for each Scene of self-testing on which Scene the model was trained for.

FIGURE 5.6: Varying the number of channels changes the generalization characteristics of a given m. Exactly which is best is uncertain, but it is unequivocally clear that each is better than TTC-Forces. Our prior results use $m = 2$, as this shows it is sufficient and seemingly the best for generalization. However, these results were tested once and would vary given more samples; nonetheless, C-TTC is better than TTC-Forces for many numbers of channels. For parameter sets tested on scenarios they were trained for, varying the amount of communication quantitively can improve the overhead by small amounts. Here, using $m = 3$ appears best, however any number of channels tested always performed better than TTC-Forces.

5.6 Communication Non-Linearity

Because every agent propagates its communication through a non-linear transform after a linear transform, the whole group of agents can be thought of as a dynamic, geometric, recurrent network of communication. In that case, changing the clamp on $\vec{c}$ would vary how efficient the simulated motion is as it directly controls the forces, too. Because clamp saturates after an input of 1, there is less control over the output forces of $\boldsymbol{M}$.

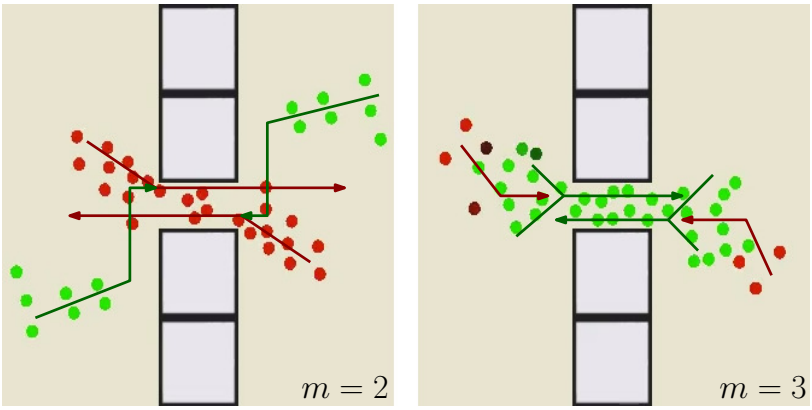


FIGURE 5.7: Visual comparison of the number of communication channels' affect on behavior. On the left is the result of training with $m = 2$ communication channels on the Doorway (D) and testing on itself. On the right is analogous the result of training with $m = 3$. Only the first communication channel is visualized, with green as positive and red as negative. This is an example of behavior that is quite common with three channels for the Doorway scene, agents will typically coordinate lanes in a more natural way. Because these are separately trained, *positive and green* means waiting on the left, but *negative and red* means waiting on the right.

Table 5.2 organizes our experimental findings with respect to the theoretical properties of a function. Here is a summary of the reasoning, and in no particular order: It is better to approximate the identity near 0. This allows the communication values to default to near-TTC when they are near-0; furthermore, this attempts to stay as similar as possible to a purely linear relationship. It is better to clamp to a maximum of 1; this is to avoid explosive propagated communication values or saturated forces. It is better to clamp to a minimum of 0 or -1; this is largely to provide some kind of gated behavior, for 0 the choice is between TTC-Forces and modified-TTC-Forces, and for -1 the choice is between modified-TTC-Forces and inversely-modified-TTC-Forces. It should be horizontally asymptotic to its ranges; this is largely to stay as consistent as possible with the idea that larger communication values should have a larger effect. It should have a slower asymptotic convergence to delay saturation of $\vec{c}$ [14].

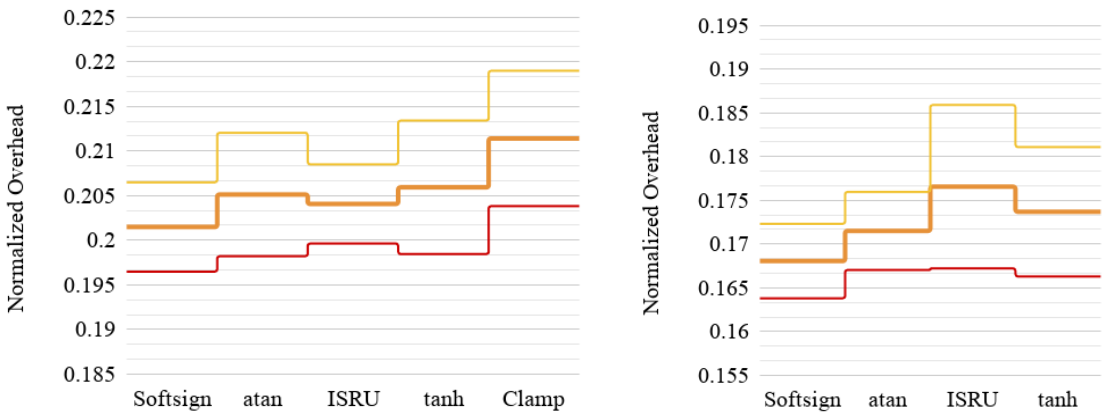
For functions which are feasible, the only remaining distinction is at what order of magnitude and rate do they converge towards their asymptote. This is literally the rate at which a function will

TABLE 5.2: Various potential non-linear functions to apply to the communication, and their properties. The theoretically ideal properties of various functions are also emphasized. The functions are roughly ordered from top-to-bottom in order of theoretically best-to-worst. The horizontal rule partitions the functions into a feasible set (top) and a non-feasible set we (bottom); only the feasible functions were tested extensively. Note, functions written with a subscript indicate their modified range (e.g, $\tanh_{0..1}$, which is computed as $\frac{f(2x)+1}{2}$).

Function	Name	${}^{df(0)}/_{dx} = 1$	Range	Range = Asymptotes	Convergence Rate Order
$\frac{x}{1+\lvert x\rvert}$	Softsign [14]	**Y**	**-1..1**	**Y**	**Linear**
$\frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right)$	atan	**Y**	**-1..1**	**Y**	**Linear**
$\frac{x}{\sqrt{1+x^2}}$	ISRU [10]	**Y**	**-1..1**	**Y**	Quadratic
$\tanh(x)$	tanh	**Y**	**-1..1**	**Y**	Exponential
$\max(-1,\min(1,x))$	Clamp	**Y**	**-1..1**	**Y**	Infinite
$\tanh\frac{x}{2}$	Shallow tanh	N	**-1..1**	**Y**	Exponential
$\max(-6,\min(6,x))$	$\text{Clamp}_{-6..6}$	**Y**	-6..6	**Y**	**Linear**
$\frac{4x}{4+x^2}$	Rational	**Y**	**-1..1**	N	**Linear**
$\max(0,x)$	ReLU	N	**0**..∞	N	n/a
$\max(.1x,x)$	Leaky ReLU	N	$-\infty..\infty$	N	n/a
$\sin x$	Sine	**Y**	**-1..1**	N	n/a

saturate towards the maximum range of communication. Figure 5.8 claims that having a function which saturates slower will typically perform better when trained and tested on the same scenario, although only marginally. Note that generating Figure 5.8 requires running the same training multiple times over which can take many hours if not days, so clamp was omitted from Figure 5.8b due it being clearly the worst in our experiments. (Table 5.2 lists the convergence rates of each function.) For a function to be approaching its limit linearly means it approaches it at the same order of magnitude as $\frac{1}{x}$ approaches 0. The meaning is similar for quadratic and $\frac{1}{x^2}$, and exponential and $\frac{1}{e^x}$. To determine the magnitude of convergence, take the limit as the function goes to infinity of the ratio of the two functions:

$$\lim_{x\to\infty}\frac{|f(x)-L|}{o(x)} \tag{5.4}$$



(A) 95% Confidence interval of means of 10 self-tested experts on Scene *F*.

(B) 95% Confidence interval of means of 5 self-tested experts on Scene *D*.

FIGURE 5.8: Testing the convergence rate of non-linearities. Left to right shows increasing convergence rates. Note that the 0 to 1 range 's result is averaged with the -1 to 1 range's result for each function.

where L is the asymptotic limit of f(x) and o(x) would be $\frac{1}{x}$ for linear, and etc. One exception is for exponential convergence, where you calculate the same function above except on a log-log scale. Because clamp always dominates the convergence rate of every function except itself, we consider it to have an "infinite" order of magnitude of convergence to 1. Furthermore, functions which do not converge, such as ReLU, do not have a convergence rate either.

All infeasible functions, typically performed on the order of twice as much overhead as the feasible functions when self-tested. For example, sin(x) consistently performed worse than TTC-Forces in nearly every transfer of an expert $\boldsymbol{M}$. It still performed better than TTC-Forces in self-testing, although with typically twice as much overhead as other functions. In general, any function that can recurrently pass into itself with convergent, saturating values is feasible for any given scenario. Figure 5.9 highlights this well in that every function performed comparably well. The difference between using 0 to 1 or -1 to 1 ranges is unclear, however, in that it appears to be function dependent when it is more useful for generalizing.

In general, the theory was drawn from whichever initial experiments worked. So, it's not surprising that further experiments on feasible functions continued to work well.Using $\text{softsign}_{0..1}$,

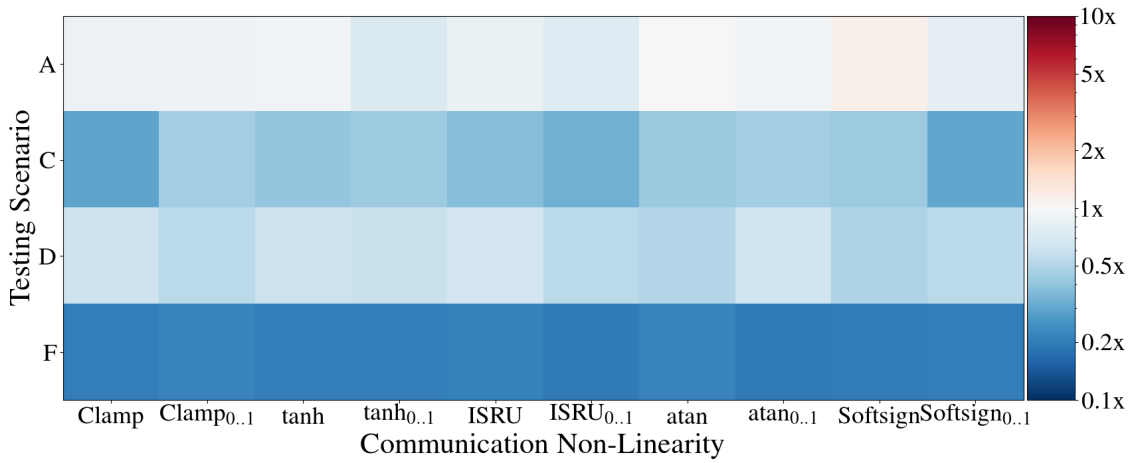


FIGURE 5.9: Average across 10 separate sessions of training on scene F and testing on four scenes by varying the non-linearity used. Note that these results use $m = 3$ communication channels.

softsign, $\text{atan}_{0..1}$, or atan seems to be the best choice as they are both linearly convergent and are non-divergent when recurrently composed for propogated communication.

5.7 Feature Selection

In Chapter 3 we used the nearest neighbor to select the features of $\boldsymbol{M}$. However, this component of C-TTC can be varied to produce different behavior, especially as related to the dynamics of the communication. Feature selection should have a significant impact on the transferrability of any given M's dynamics, as it is the core feature to provide invariance to the number of agents and their density. However, here, we focus on how the selection functions change C-TTC's behavior under optimized conditions. Figure 5.10 compares four selection methods for three different non-linear communication functions when they are trained and tested on Scene D.

Volume selections attempt to pick agents which represent the aggregate communication of the nearby group:

$$j = \underset{j \in A}{\text{argmax}} \left(\frac{\sum_m |\vec{c}_{jm}|}{\|\vec{p}_j - \vec{p}_i\|^2} \right) \tag{5.5}$$

But varied with using $\text{Softsign}_{0..1}$ for the non-linearity, Softsign, or no absolute value on $\vec{c}_j$ with Softsign, which is a “negative" volume selection. Furthermore, argmax would be replaced with an argmedian for median selection. Max volume selection will pick the “loudest" agents, which generally leads to very saturated communication and forces. In contrast, median volume selection will take the “typical" agent, which tends to regularize the communication and behavior towards TTC-Forces. This can be seen for both feature selections methods for both $\text{Softsign}_{0..1}$ and Softsign in Figure 5.10.

The purpose of a negative volume selection is to allow non-linear communication functions with a range of -1 to 1 to emulate those with a range of 0 to 1, as it will have a similar gating effect on whether TTC-Forces is being modified or not. Furthermore, it will have a tendency to regularize the communication values. If not modifying TTC-Forces is optimal, then weighting too negatively will result in saturated forces once an agent with large negative values is selected. Because these forces may lead to non-optimal behavior, models which produce large negative values would not be optimized towards. This regularizing effect can be seen for maximum volume selection in Figure 5.10, where most agents have a neutral-yellow color, with only a few standing out to communicate.

As an alternative to volume selection methods, agents can take the weighted sum of their coordination forces before applying any clamping or non-linearity:

$$\begin{aligned} \vec{c}_i' &= \sigma\left(\sum_j \frac{\begin{bmatrix}\boldsymbol{M}_c & \boldsymbol{M}_o\end{bmatrix} \cdot \begin{bmatrix}\vec{c}_j & \vec{o}_j\end{bmatrix}^T}{\|\vec{p}_j - \vec{p}_i\|^2}\right) \\ \vec{F}_{ci} &= \textsc{Clamp}\left(\sum_j \frac{\boldsymbol{M}_F \cdot \vec{c}_j}{\|\vec{p}_j - \vec{p}_i\|^2}\right) \end{aligned} \tag{5.6}$$

The advantage of taking the distance weighted sum of each agent's contribution is that it can provide a global cohesion where all agents move as one. The disadvantage to this method is that its no longer truly invariant, as denser scenarios will result in more saturation of the communication and coordination force.

We exclude selection methods which had large overheads, which was largely any method that did not weight nearby agents higher. Selection methods which do not weight distance, such as

selecting j to be the agent with the maximum $\sum |vecc|$, can no longer respond to local congestion as effectively.

Each of the selection methods have less overhead than TTC-Forces by each forming efficient lanes. Five of the ten trials shown have nearly the same or symmetric versions of the same lane formations; this is likely due to a local minima of optimization, which partly suggests that selection has little effect on an expert model's behavior, but its communication dynamics would still affect its transferrability. Furthermore, in every example shown, because $m = 3$, none of the waiting groups are attempting to cross over lanes. Of interest are the five methods which behaved uniquely, two of which in particular stand out: median volume selection using $\text{Softsign}_{0..1}$ and nearest neighbor using Softsign. For the nearest neighbor selection, nearly every agent waited except for a few that formed lanes, after which the rest quickly cram into the fast-moving lanes. In contrast, the median selection's agents probed with yellow colored agents to form lanes while the rest followed closely. The more dynamic use of communication in the median selection is probably due to both the $\text{Softsign}_{0..1}$ and the median volume, as the former allowed black (TTC-Forces) agents to appear and the latter regularized the rest of the communication values to not deviate far unless necessary.

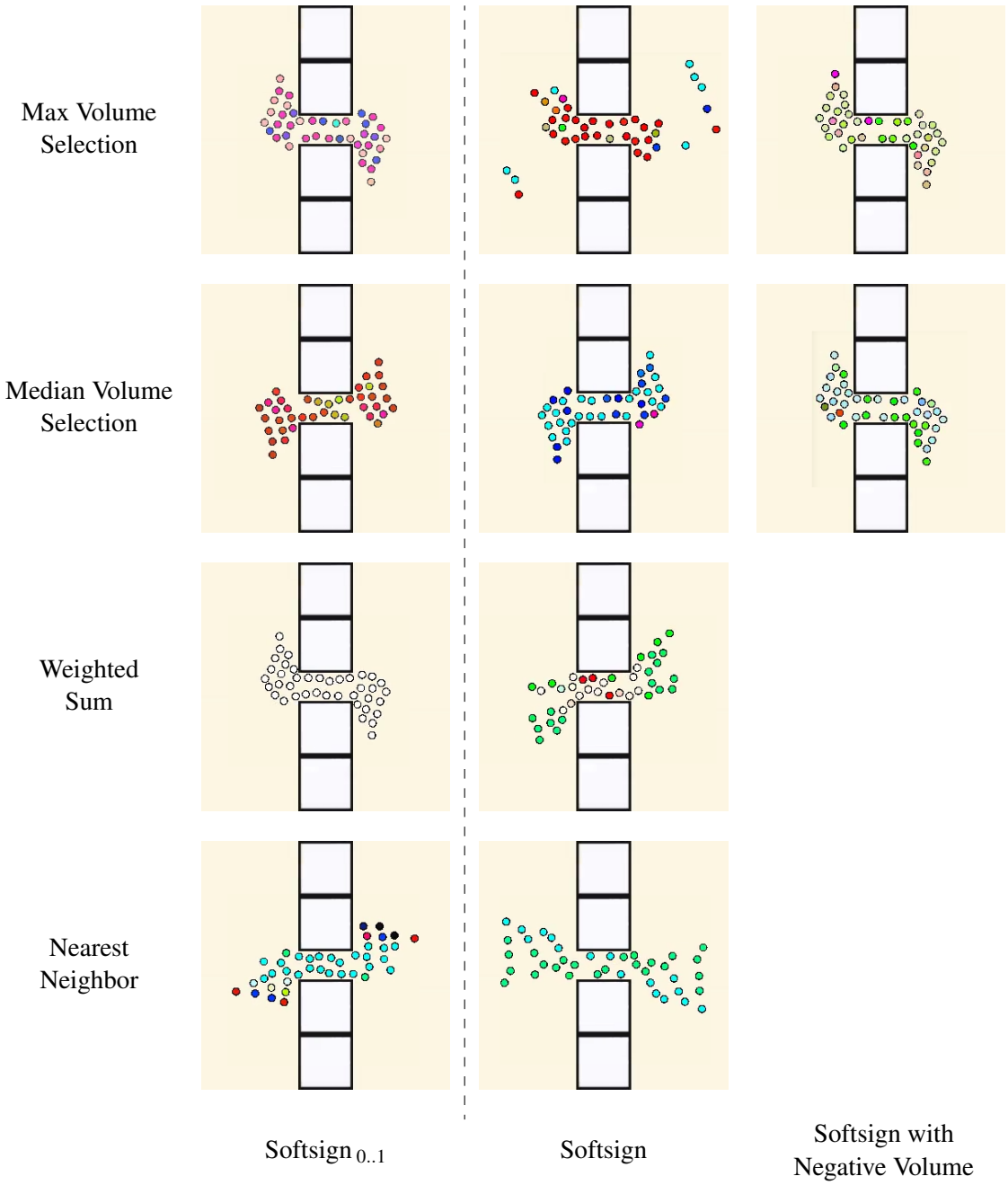


FIGURE 5.10: Comparison of linguistic behavior and motion from changing the selection of which agents are listened to. This shows snapshots five seconds into the self-testing of the Scene D expert. Note that each of these were trained using $m = 3$ communication channels.

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 5.2 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 ways 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.

6.1 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 [41] or ORCA [6]. 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 [25]. 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.

C-TTC does not use a *grammatical* language of communication between agents, even though its policies have some linguistic properties. For example, there is no concept of hierarchy or abstraction within the communication policy. Hierarchical communication values would be useful for agents to propagate and exchange higher-order plans, or even reason over their dynamic, geometric network, as would be required for any agents needing to re-plan instead of re-navigate as they solely do in C-TTC. A concrete example would be if agents could route around a building once it has been communicated that a large number of agents are currently planning on exiting the building. Furthermore, other symbolic abstractions are not present in C-TTC's communication policies, such as "true" silence, which is only expressed by the static obstacles of the environment, and would be useful for ignoring spatial observations. Ideally, such abstractions could emerge from optimization as, like with the communication policies, there is a wide variety of possible scenarios to account for.

6.2 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, although, we've shown that many components of C-TTC can be varied for meta-optimizations. In this regard, we are also excited about approaches to multi-scene optimization that consider the Pareto-frontier, as this would alleviate much of the regularizing effect caused by "simpler" scenarios. We are particularly excited about drawing on ideas successful from reinforcement learning and neural networks. For example, using neural networks for each agent's parameters instead of a linear transforms or training the parameters of $\boldsymbol{M}$ with reinforcement learning algorithms, such

as A3C [32]. However, there are some added difficulties for applying internal neural networks to C-TTC; most notably, the optimization would be much more difficult. There would only be one gradient per simulation that would have to back-propagate through all of the communication non-linearities during the entire simulation as well as any internal neural networks. 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 parameter set we use now. There are two apparent methods that $\boldsymbol{M}$ could be varied, either it could be directly interpolated as the output of another internal neural network or a separate algorithm could form a consensus among agents on when to discretely change experts. The former approach has the advantage of being decentralized and its able to mix its set of experts. However, it would run into discontinuities in the evaluation space that could lead to much worse behavior. The latter approach has the advantage of consistently using the norms of communication between agents; however, it may lead to a disjoint grouping of agents when there is not a full consensus, which might cause global inefficiencies again.

BIBLIOGRAPHY

[1] Javier Alonso-Mora et al. “Image and animation display with multiple mobile robots”. In: *The International Journal of Robotics Research* 31.6 (2012), pp. 753–773.

[2] Tucker Balch and Ronald C Arkin. “Communication in reactive multiagent robotic systems”. In: *Autonomous robots* 1.1 (1994), pp. 27–52.

[3] O Burchan Bayazit, Jyh-Ming Lien, and Nancy M Amato. “Better Group Behaviors in Complex Environments using Global”. In: *Artificial Life 8* 8 (2003), p. 362.

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

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

[6] Jur van den Berg et al. “Reciprocal n-body collision avoidance”. In: *Robotics research* (2011), pp. 3–19.

[7] Glen Berseth et al. “SteerFit: Automated parameter fitting for steering algorithms”. In: *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. Eurographics Association. 2014, pp. 113–122.

[8] Graeme Best et al. “Planning-aware communication for decentralised multi-robot coordination”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1050–1057.

[9] PC Buzing, AE Eiben, and Martijn C Schut. “Emerging communication and cooperation in evolving agent societies”. In: *Journal of Artificial Societies and Social Simulation* 8.1 (2005).

[10] Brad Carlile et al. “Improving deep learning by inverse square root linear units (ISRLUs)”. In: *arXiv preprint arXiv:1710.09967* (2017).

[11] Jakob Foerster et al. “Learning to communicate with deep multi-agent reinforcement learning”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 2137–2145.

[12] Dieter Fox et al. “Distributed multirobot exploration and mapping”. In: *Proceedings of the IEEE* 94.7 (2006), pp. 1325–1339.

[13] Mohammad Ghavamzadeh and Sridhar Mahadevan. "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. 2004, pp. 1114–1121.

[14] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[15] Julio Godoy et al. "ALAN: adaptive learning for multi-agent navigation". In: *Autonomous Robots* (2018), pp. 1–20.

[16] Julio Erasmo Godoy et al. "Implicit coordination in crowded multi-agent navigation". In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[17] Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. "Coordinated reinforcement learning". In: *ICML*. Vol. 2. Citeseer. 2002, pp. 227–234.

[18] Stephen J Guy and Ioannis Karamouzas. "Guide to Anticipatory Collision Avoidance". In: *Game AI Pro 2*. Ed. by Steve Rabin. CRC Press, 2015. Chap. 19, pp. 195–208.

[19] Stephen J Guy et al. "Pledestrians: a least-effort approach to crowd simulation". In: *Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on computer animation*. Eurographics Association. 2010, pp. 119–128.

[20] Dirk Helbing, Illés Farkas, and Tamas Vicsek. "Simulating dynamical features of escape panic". In: *Nature* 407.6803 (2000), p. 487.

[21] Dirk Helbing and Peter Molnar. "Social force model for pedestrian dynamics". In: *Physical review E* 51.5 (1995), p. 4282.

[22] Suranga Hettiarachchi. "An evolutionary approach to swarm adaptation in dense environments". In: *ICCAS 2010*. IEEE. 2010, pp. 962–966.

[23] Seung Wan Hong, Davide Schaumann, and Yehuda E Kalay. "Human behavior simulation in architectural design projects: An observational study in an academic course". In: *Computers, Environment and Urban Systems* 60 (2016), pp. 1–11.

[24] Mubbasir Kapadia et al. "Multi-domain real-time planning in dynamic environments". In: *Proceedings of the 12th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. ACM. 2013, pp. 115–124.

[25] Ioannis Karamouzas, Brian Skinner, and Stephen J Guy. "Universal power law governing pedestrian interactions". In: *Physical review letters* 113.23 (2014), p. 238701.

[26] Ioannis Karamouzas et al. "A predictive collision avoidance model for pedestrian simulation". In: *International Workshop on Motion in Games*. Springer. 2009, pp. 41–52.

[27] Ali Abdul Khaliq and Alessandro Saffiotti. "Stigmergy at work: Planning and navigation for a service robot on an RFID floor". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 1085–1092.

[28] Andrew Kimmel, Andrew Dobson, and Kostas Bekris. "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. 2012, pp. 247–256.

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

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

[31] Francisco S Melo and Manuela Veloso. "Learning of coordination: Exploiting sparse interactions in multiagent systems". In: *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems. 2009, pp. 773–780.

[32] Volodymyr Mnih et al. "Asynchronous methods for deep reinforcement learning". In: *International conference on machine learning*. 2016, pp. 1928–1937.

[33] Soraia Raupp Musse and Daniel Thalmann. "Hierarchical model for real time simulation of virtual human crowds". In: *IEEE Transactions on Visualization and Computer Graphics* 7.2 (2001), pp. 152–164.

[34] Brammert Ottens and Boi Faltings. "Coordinating agent plans through distributed constraint optimization". In: *Proceedings of the ICAPS-08 Workshop on Multiagent Planning*. 2008.

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

[36] Lynne E Parker. "ALLIANCE: An architecture for fault tolerant multirobot cooperation". In: *IEEE transactions on robotics and automation* 14.2 (1998), pp. 220–240.

[37] Nuria Pelechano and Norman I Badler. "Modeling crowd and trained leader behavior during building evacuation". In: *IEEE computer graphics and applications* 26.6 (2006), pp. 80–86.

[38] Fasheng Qiu and Xiaolin Hu. "Modeling group structures in pedestrian crowd simulation". In: *Simulation Modelling Practice and Theory* 18.2 (2010), pp. 190–205.

[39] Matt Quinn. "Evolving communication without dedicated communication channels". In: *European Conference on Artificial Life*. Springer. 2001, pp. 357–366.

[40] Zhiguo Ren et al. "Group Modeling: A Unified Velocity-Based Approach". In: *Computer Graphics Forum*. Vol. 36. 8. Wiley Online Library. 2017, pp. 45–56.

[41] Craig W Reynolds. "Flocks, herds and schools: A distributed behavioral model". In: *ACM SIGGRAPH computer graphics* 21.4 (1987), pp. 25–34.

[42] Matthew Schuerman et al. "Situation agents: agent-based externalized steering logic". In: *Computer Animation and Virtual Worlds* 21.3-4 (2010), pp. 267–276.

[43] Sainbayar Sukhbaatar, Rob Fergus, et al. “Learning multiagent communication with backpropagation”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 2244–2252.

[44] Vincent Tabak, Bauke de Vries, and Jan Dijkstra. “Simulation and validation of human movement in building spaces”. In: *Environment and Planning B: Planning and Design* 37.4 (2010), pp. 592–609.

[45] Ioan Cristian Trelea. “The particle swarm optimization algorithm: convergence analysis and parameter selection”. In: *Information processing letters* 85.6 (2003), pp. 317–325.

[46] Branislav Ulicny and Daniel Thalmann. “Towards Interactive Real-Time Crowd Behavior Simulation”. In: *Computer Graphics Forum*. Vol. 21. 4. Wiley Online Library. 2002, pp. 767–775.

[47] Sai-Keung Wong et al. “Guidance path scheduling using particle swarm optimization in crowd simulation”. In: *Computer Animation and Virtual Worlds* 26.3-4 (2015), pp. 387–395.

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

[49] Hengchin Yeh et al. “Composite agents”. In: *Proceedings of the 2008 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. Eurographics Association. 2008, pp. 39–47.

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