2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

When, What, and with Whom to Communicate: Enhancing RL-based Multi-Robot Navigation through Selective Communication

Senthil Hariharan Arul[1], Amrit Singh Bedi[2], and Dinesh Manocha[1]

Abstract— **Decentralized navigation methods rely primarily on local observations, lacking the global awareness needed to coordinate effectively within a multi-agent system. Exchanging relevant messages between agents can promote cooperation and improve navigation efficiency. We present a Reinforcement Learning (RL)-based decentralized navigation approach that learns '*when*,' '*what*,' and '*with whom*' to communicate for safe and cooperative navigation. Our method leverages a visual transformer and self-attention mechanism to encode the local occupancy map and the state information of neighbors into fixed-length encodings, allowing it to handle an arbitrary number of neighbors for collision-free navigation. In addition, the network encodes the agent's state information and observations of neighboring agents into a concise message vector by learning *what* information is crucial to communicate, which is shared with neighboring agents upon request. Moreover, to avoid indiscriminate broadcasting, the network learns *when* and *with whom* to communicate and request message vectors. Subsequently, the messages communicated alongside the local information are used to guide navigation decisions. We evaluate our method against state-of-the-art baselines in complex scenarios, including narrow corridors and environments with multiple agents. We observe considerable improvements in terms of navigation performance, showing up to $\sim 2\times$ improvement in navigation success rates and a reduction of up to $\sim 20\%$ in path length.**

I. Introduction

Recent advancements in perception, planning, and deep learning have driven robots into multiple novel real-world applications. In these contexts, robots play a pivotal role in enhancing productivity, precision, reliability, and safety across different domains. For instance, robots excel at efficiently moving payloads within warehouses, conducting inspections in hazardous environments to ensure human safety, or surveilling areas for security purposes. Moreover, many such applications benefit from utilizing multiple robots in tandem [1].

Irrespective of their specific application, robots within multi-robot systems must navigate between locations safely and efficiently to fulfill their objectives. Multi-robot navigation methods can be broadly categorized into centralized [2], [3] and decentralized methods [4], [5], [6]. Centralized approaches involve a central system computing a plan for all robots in the system to guide them to the goal and avoid collisions. While they can achieve optimal plans, they are not scalable to a large number of agents due to the exponential increase in the number of states [7], [8].

This work is supported in part by ARO Grants W911NF2310046, W911NF2310352 and US Army Cooperative Agreement W911NF2120076. We acknowledge the support of the Maryland Robotics Center.

[1]Authors are with the University of Maryland, College Park, MD 20740, USA. {sarul1,dmanocha}@umd.edu

[2]Author is with University of Central Florida, FL, USA amritbedi@ucf.edu

In decentralized methods, agents independently decide on actions based on their observations of the local environment. While decentralized approaches offer scalability and resilience through distributed decision-making, they may generate suboptimal plans and lead to deadlocks or even collisions in complex environments. One contributing factor to these challenges is that decentralized methods, unlike centralized approaches, depend on observable state information of neighboring robots and lack insight into their goals or navigation intentions. While complete knowledge about all robots could be advantageous, it is unrealistic to assume continuous access to such knowledge.

Similarly, human navigation involves individuals making independent decisions based on locally observed information. Despite this, human navigation is quite efficient, capable of resolving conflicts and exhibiting emergent crowd behaviors, such as forming lanes for efficiency [9]. Humans further enhance their navigation by communicating their intentions to neighbors. For example, drivers use turn signal indicators while driving, and pedestrians may directly request someone blocking a doorway to move.

Recently, learning-based approaches [10], [11], [12] have been used for decentralized planning. In these methods, the network learns cooperative navigation by simulating the intricate interactions among multiple agents during training, while relying solely on local information. Furthermore, researchers have explored communication channels for the exchange of additional information or messages. This augmentation has shown promise in enhancing navigation performance in multi-robot systems with partial observation, resulting in improved safety, deadlock resolution, and other desirable outcomes [13], [14], [15]. However, existing approaches either focus on solving the problem of planning over a graph (multi-agent pathfinding) or involve communicating planned actions over a predetermined time horizon [14], without addressing the crucial question of determining what information is most beneficial to communicate.

Hence, enabling robots to exchange relevant information with others can aid coordination in decentralized multi-robot systems and improve their navigation performance. However, real-world communication faces challenges such as limited bandwidth, transmission delays, and potential security issues. Therefore, it's necessary to communicate intelligently and only when necessary to improve navigation. Furthermore, the communicated information could include robot intentions, observations, goal information, or planned future trajectories. Hence, the critical challenge lies in determining what infor-

mation is communicated, when, and to whom, to improve overall navigation performance.

A. Main Contributions

In this paper, we address the problem of navigating multiple robots through complex environments riddled with static obstacles while ensuring collision-free navigation in a decentralized fashion. Specifically, we propose an RL-based approach that focuses on the relatively underexplored realm of simultaneously learning navigation strategies and efficient communication strategies—deciding when, what, and whom to communicate with—to enhance navigation performance using reinforcement learning techniques. Our main contributions are:

- We propose a novel RL-based approach for multi-robot navigation that leverages communication to enhance navigation performance. Our network simultaneously learns navigation and efficient communication, including determining what message to communicate and whom to communicate with, to improve navigation based on the applied reward.
- Our network encodes the local information of the agent, encompassing neighbor state data, ego-state, and local occupancy map, into a concise, fixed-length message vector that neighboring agents can request. Furthermore, our network uses attention to learn an encoding of the neighbor's observable state, which is then utilized to predict the probability of a communication link to initiate communication with the specific neighbor.
- We evaluate the efficacy of our method across various challenging environments, with static obstacles and multiple agents, by benchmarking against state-of-the-art learning and non-learning baselines. We observe that the proposed method significantly improves key metrics such as path length (up to $\sim 20\%$), and success rate (up to $\sim 2\times$) compared to the baselines.

Moreover, we evaluate the selective communication performance by presenting the number of communication link made in different scenarios in relation to broadcast communication. Additionally, we present a discussion through correlation analysis, highlighting the influence of inter-agent distance and agent velocity on selective communication.

II. Related Works

Reactive methods [4], [16] are a class of decentralized methods that use the current state of the agents to inform navigation decisions. Velocity obstacles (VO) [4] is a popular reactive method that computes a set of collision-free velocities between pairs of robots. To improve coordination, RVO [17] extended the VO concept by accounting for the collaborative effort between the agents to avoid collision. ORCA [18] approximates velocity obstacles as half-planes and formulates the collision avoidance problem as a linear programming problem. While ORCA scales to large multi-agent systems, it can result in deadlocks in dense scenarios. In contrast to planning in velocity space, BVC [5] plans in position space by Voronoi cells are free space. In [19], the authors present a hybrid approach based on BVC and velocity obstacles for multi-agent navigation and deadlock resolution. Other methods make use of potential field [20] and control barrier function [21] for collision avoidance. Another class of methods based on the model predictive controller (MPC) [22], [6] make use of a finite time horizon prediction of neighbor trajectories to find a collision-free, smooth trajectory to the goal. Even with perfect sensing capabilities, decentralized methods can still lead to deadlocks, collisions, and other issues in the absence of cooperation [23].

Data-driven approaches that model human crowd navigation in terms of steering efficiency [24] and navigation interaction [25] have been explored. For instance, in [25] the authors proposed that collision avoidance between humans in a crowd follows a power law relation on the estimated time-to-collision. NH-TTC [26] used this idea for collision avoidance among agents with arbitrary equations of motion.

Learning-based methods model the complex interaction between agents during a training process, which is used during execution to compute a suitable action for the agent. Despite lacking formal safety guarantees, or explainability, they have received tremendous focus from the research community due to their impressive empirical performance against other state-of-art baselines. Deep reinforcement learning methods [10], [11] have been explored that learn multi-agent navigation between non-communicating scenarios from applied reward. In [12], the method accounts for an arbitrary number of neighboring agents by encoding neighbor information using LSTM.

A. Multi-Agent Communication

Prior works [27], [28], [29], [30] have considered the benefits of communication for multi-agent systems. Particularly, Balch et al. [27] explored user-designed direct communication with assigned meanings. Godoy et al. [28] explore exchanging specific information such as goal and velocity information for more coordinated navigation. While these methods validate the intuition that communicated information beyond observed neighbor states can enhance coordination, broadcasting with all agents may be infeasible due to bandwidth limitations, communication delays, and other factors. A DRL method for multi-agent pathfinding with broadcast communication is presented in [13]. The approach in [31] uses local communication to coordinate agent motion where they use machine learning to learn the communication message. Furthermore, communicating with every agent indiscriminately (as in broadcast) may be unnecessary, as not all agents are relevant and impact an agent's navigation decision. Therefore, selectively communicating relevant messages between agents becomes imperative.

Boardman et al. [30] define a rule-based method for sporadic communication in a multi-agent system. Das et al. [32] present RL-based targeted multi-agent communication approach. A few works have explored learning communication in the multi-agent navigation domain. Serra-Gomez et al. [14] learn with whom to communicate and

Symbols	Description
$\mathbf{p}_i$	Position of agent i (p_x, p_y)
$\mathbf{g}_i$	Goal of agent i (g_x, g_y)
ψ_i	Heading of agent i
r_i	Radius of agent i
$\mathbf{v}_i$	Velocity of agent i (v_x, v_y)
v_{pref_i}	Preferred Velocity of agent i
m^{occ}	Occupancy Map of agent i
$\mathcal{N}_i$	Set of neighbors of agent i
$\mathcal{C}_i$	Set of selected neighbors for communication

TABLE I: Symbols and notations

request the planned trajectories from the chosen neighbor to be used in the MPC planner. Ma et al. [33], present an approach based on inferred communication [34] to reduce the communication overhead for multi-agent navigation in a grid world domain. In DMCA [15], authors present an RL-based method of learning navigation and selective communication in the non-grid world setting, where the goal information gets communicated with relevant neighbors. We use a similar attention based mechanism to encode the neighbor information. In SelComm [35], each agent selects relevant neighbors to receive their message, with each neighbor broadcasting their information. However, in applications such as military scenarios, constant broadcasting of information may not be ideal due to security concerns.

In contrast to the above works, our approach uses reinforcement learning (RL) to jointly learn selective communication and the message to be communicated alongside navigation. We employ a request-reply communication scheme, where the communication gets initiated by the agent receiving the message. When requested, the neighbor shares a concise message vector.

III. Preliminaries

In this section, we outline our problem statement, assumptions, and other relevant preliminaries for our approach. Additionally, Table I summarizes the variables and notations frequently referenced throughout our paper.

A. Problem Formulation: Multi-Agent Navigation

Consider a team of N disk-shaped robots operating in a shared environment $\mathcal{W} \subset \mathbb{R}^2$. Let $\mathcal{X}_{free} \subset \mathcal{W}$ represent the free space, and $\mathcal{X}_{obs} \subset \mathcal{W}$ represent the static obstacle space. Each robot's geometry is represented by $\mathcal{A}_i$, where $i \in \{1, 2, \cdots, N\}$, with its 2D position and goal given by $\mathbf{p}_i \in \mathcal{X}_{free}$, and $\mathbf{g}_i \in \mathcal{X}_{free}$, respectively. Besides, we consider that robots are capable of communicating with their neighbors. For each time step t, we can mathematically express the safe navigation problem for robot i as follows:

$$\arg\min_{\pi} \quad ||\mathbf{p}_i(t) - \mathbf{g}_i|| \tag{1}$$

$$\mathcal{A}(\mathbf{p}_i(t)) \cap \mathcal{A}(\mathbf{p}_j(t)) = \varnothing \quad \forall i, j, \ \ i \neq j \ \ \forall t \tag{2}$$

$$\mathcal{A}(\mathbf{p}_i(t)) \cap \mathcal{X}_{obs} = \varnothing \tag{3}$$

Here, $\mathbf{p}_i(t)$ represents the position of the robot i at time t. The objective is to learn a policy π to navigate the robot, that minimizes the robots distance to the goal (1) while maintaining a safety distance from other agents (2) and obstacles (3).

B. Assumption

In this work, we assume disk-shaped robots and that any pair of robots can communicate with each other. The communication follows a request-reply framework, that is, an agent requests its neighbor for information and the neighbor responds with the message vector. We assume this inter-agent communication is fast and is executed within the planning cycle, and it is assumed that the neighbor will always respond with the requested message.

C. Robot State

Following the definition from [10], the agent's state $\mathbf{s}_i = [\mathbf{s}_i^o, \mathbf{s}_i^h]$ comprises a component observable by its neighbors ($\mathbf{s}_i^o$) and a component hidden to them ($\mathbf{s}_i^h$). The observable component $\mathbf{s}_i^o$ consists of the agent's position, velocity, and radius, while the hidden component $\mathbf{s}_i^h$ includes the agent's goal, preferred speed, and the current orientation. Therefore, we represent them as: $\mathbf{s}_i^o = [p_x, p_y, v_x, v_y, r], \mathbf{s}_i^h = [g_x, g_y, v_{pref}, \psi]$.

Using this, the ego-agent state for network input includes its distance to the goal, preferred speed, orientation, and radius, and is given by

$$\mathbf{s}^{ego} = [||\mathbf{g}_i - \mathbf{p}_i||, \ \ v_{pref_i}, \ \ \psi_i, \ \ r_i]. \tag{4}$$

The neighbor state information for an ego-agent includes their positions ($\tilde{\mathbf{p}_j}$) , velocities ($\tilde{\mathbf{v}_j}$) w.r.t the ego frame, radii, inter-agent distances (d_a), and combined radii denoted as

$$\mathbf{s}^{obs_j} = [\tilde{p_{x_j}}, \ \tilde{p_{y_j}}, \ \tilde{v_{x_j}}, \ \tilde{v_{y_j}}, \ r_j, \ d_a, \ r_i + r_j]. \tag{5}$$

In addition, in our approach, we include the occupancy map information and communicated message as an input to the network for navigation.

1) Occupancy Map (m^{occ}) : We use an occupancy grid map to represent the static obstacle information in the environment. The robot is centered on the occupancy map, and the orientation of the map is aligned with the robot's orientation.

2) Communicated States: Moreover, an ego agent can request its neighbors for information to improve its decision-making. To enable this, each agent encodes its state information in a fixed-length vector, which can be shared with the neighbor upon request.

D. Action Space

Similar to [10], we consider speed and change in angle as actions, i.e., $\mathbf{u}_i = [v_i, \Delta\psi_i]$. Our action space is set of 27 actions varying based on speed and a change in angle. Here, $\Delta\psi_i \in \{0, \pm\frac{\pi}{24}, \pm\frac{\pi}{12}, \pm\frac{\pi}{8}, \pm\frac{\pi}{6}\}$ and $v_i \in \{0.0, 0.5 \cdot v_{pref}, v_{pref}\}$.

E. Vision Transformer

Vision transformers (ViTs) [36] are transformer architectures employed for image processing tasks. They segment an image into fixed-size patches, which are then linearly transformed to generate a sequence of embeddings for a transformer model. In this paper, we leverage a ViT to encode the input occupancy grid map.

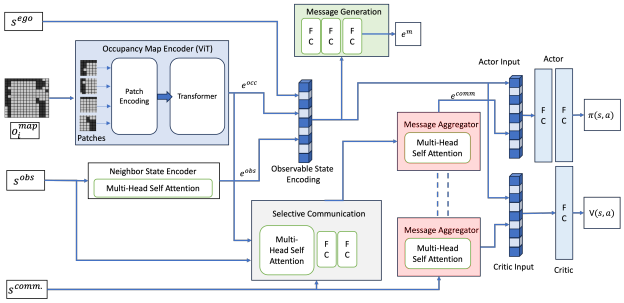


Fig. 1: Network Architecture: Our network takes the local occupancy map (o^{map}), the agent's state (s^{ego}), and the observable states of the neighbors (s^{obs}) as inputs. The occupancy map and neighbor state inputs are encoded and combined with the agent's state to form the observable state encoding. This encoding serves as input for message generation and for the actor and critic networks. Additionally, the message received through selective communication is encoded into a fixed-length vector by the message aggregator. Furthermore, for the critic input, an encoding of messages from all neighbors is utilized. The message aggregator, employed for both critic input and for aggregating selected messages for actor input, shares the network weights.

IV. Network Architecture

At a high level, our network leverages local environmental information, including the occupancy map, its state, and neighbors' observable state information, to guide its navigation decisions. In addition to this local information, the network selects relevant neighbors by learning whom to request messages to enhance navigation. In this section, we describe our network architecture and reward structure. Figure 1 illustrates our network architecture.

A. Occupancy Map Encoding

Our network employs a binary occupancy grid map (m^{occ}) of $3m \times 3$m to represent the static obstacles within the local environment. Each cell in the occupancy grid map is represented by $m^{occ}(u, v)$. The grid cell denoting free space is assigned a value of 0, whereas those indicating an obstacle are assigned a value of 1. We apply distance transformation to the binary occupancy map, converting it to an output map with each grid cell containing the distance (in the grid cell) to the closest obstacle cell. We represent the resulting map by $d_{obs.}(m_{occ})$. Subsequently, we transform the value of each grid cell to a continuous value between 0 and 1 using the following exponential function, which can be seen as a measure of collision probability [37]. Thus, the resulting occupancy map

$$o^{occ}(u, v) = \exp - \frac{d_{obs.}(m^{occ\cdot}(u, v))^2}{\sigma^2_{obs.}} \quad \forall (u, v) \in m^{occ}$$

Here, $\sigma_{obs.}$ represents a user-defined constant dictating the rate at which the value transitions between 0 and 1 in response to the distance from an obstacle.

We utilize a visual transformer (ViT) to encode this occupancy map information into a fixed-length representation. Our occupancy map, comprising 30×30 cells, is divided into 25 patches, each measuring 6×6 cells. Each patch undergoes linear transformation, projecting it into a 128-dimensional vector—a compact encoding of the patch.

These encoded patch vectors serve as the input for the Transformer encoder, which comprises two transformer blocks, each with 16 attention heads. Subsequently, we apply global average pooling to the transformer output to get an encoded vector representing the occupancy map information, represented by $\mathbf{e}^{occ}$.

B. State Encoding

At any given time step, the number of agents within the sensing radius of the ego-agent can vary. To accommodate this variability, we employ multi-head self-attention to encode the observed state information from all neighboring agents into a fixed-length representation. The set of neighbors for an agent i is given by $\mathcal{N}^i = \{j \quad | \quad j \neq i, \ ||\mathbf{p}_j - \mathbf{p}_i|| < r_{neighbor}\}$, The input sequence comprises the observable states of its neighbors, denoted as $\mathbf{s}_i^{obs}$ for each neighbor i in the set $\mathcal{N}$, followed by the observable state computed for ego agent relative to itself $\mathbf{s}_{ego}^{obs}$.

Each element in the sequence is projected into three 128-dimensional input vectors, serving as the input for the query, key, and value in the multi-head self-attention module. We use a self-attention layer with 10 heads and compute an encoding for each vector in the input sequence by attending to other vectors in the sequence. The encoded vector corresponding to the ego-state vector is used as the encoded representation of the observed neighbor information. The neighbor encoding is represented by $\mathbf{e}^{obs}$.

The ego agent considers the robot's state ($\mathbf{s}^{ego}$), its neighbors' state, and the static obstacles to compute a suitable action. The $\mathbf{s}^{ego}$, $\mathbf{e}^{obs}$, and $\mathbf{e}^{occ}$ together result in a fixed-length vector that represents the local information of the ego agent, which we refer to as the observed state encoding.

C. What to communicate: Message Generation

In decentralized methods, each agent has access to observable state information of its neighbors, such as their position, velocity, and radius, which can be obtained through sensing. However, they generally lack information about their neighbor's navigation intent, including the neighbor's goal information, their preferred velocity, and their local environments comprising other agents and static obstacles. This information is crucial for understanding the neighbor's motion intent and, in turn, influences the navigation decision of the ego agent. Therefore, the message communicated by an agent should effectively encapsulate its state information to assist the receiving neighbor in making informed decisions.

Hence, to generate a relevant message to communicate with neighbors, we encode the current state information of the agent, containing its occupancy map, observed neighbor states, and both observable and hidden components of its ego state, into a concise vector of size 32 ($\mathbf{e}^m$). This input information is in the ego-centric frame of the robot and is transformed through three fully connected layers to produce a concise fixed-length representation, this transformation is learned during training from the reward structure. Consequently, the message acts as an abstraction of the agent's

own state and its local surroundings, and can be shared with neighboring agents upon request. In our implementation, the message from the previous time step is communicated.

D. Message Aggregation

Let $\mathcal{C}_i$ denote the set of neighbors from which the ego-agent has requested the message, and let sm_j denote the neighbor messages, where $j \in \mathcal{C}_i$. Upon receiving messages from its neighbors, the ego-agent encodes all messages into a fixed-length representation. Since the communicated messages originate from the local frame of each neighbor, we augment each message vector with the observed state information of the respective neighbor in the ego-agent frame, resulting in $[s_j^{obs}, e_j^m]$. These augmented message vectors are then processed through three dense layers to create an encoding, serving as the query, key, and value inputs to the self-attention layer. This sequence of augmented message vectors is then processed through a message aggregator based on self-attention. The resulting output encoding represented by e^{comm}, is an abstraction of all received messages from the neighbors.

E. Whom to Communicate: Selective Communication

To enhance communication efficiency, each agent autonomously decides when and which agent to communicate with at each time step. To identify relevant neighbors for communication, we use an encoding of the robot's local environmental information and formulate the selective communication problem as a link prediction task. Specifically, we compute an encoding of each neighbor's observed state information through self-attention, then augmenting it with the occupancy map encoding $\mathbf{o}^{obs}$ computed earlier. Subsequently, this augmented encoding for each neighbor is passed through a sequence of two fully connected layers. The output of the final layer consists of two nodes, with softmax serving as the activation function. This final output provides the probability of a communication link being relevant for navigation, where one node indicates the probability of a link (p_{link}) and the other node denotes $1 - p_{link}$.

Hence, for all neighbors, the ego agent i predicts the probability of a communication link. We represent this set of selected neighbors for communication as $\mathcal{C}_i$, where $\mathcal{C}_i \subset \mathcal{N}_i$. The agent i sends a communication request to the selected neighbors $j \in \mathcal{C}_i$ and receives the generated message from the neighbor. The received sequence of messages can be represented as, $\{\mathbf{e}_j^m\} \quad \forall j \in \mathcal{C}_i$.

F. Critic Input

The critic network uses the local information of the robot, i.e. the observed state encoding, and in addition, the information from all neighboring robots to learn the value function. When training with a large number of agents, the state information from all the agents could make the input to the critic very large and training slower. To reduce this, the critic receives an encoded version of the neighbors' state information (generated messages, i.e., $\{e_j^m\}\ \forall j \in \mathcal{N}_i$), which is then converted to a fixed-length encoding with the message aggregator to be used as the critic's input. The message aggregator shares its network weights with the aggregator used to encode the selectively communicated messages.

G. Reward Structure

Our reward structure is designed to incentivize desired behavior by positively reinforcing goal attainment while discouraging collisions, excessive communications, and jerky motion. Given the complex navigation problem we consider, it is challenging to learn a good policy with sparse rewards. Therefore, we use a mixture of sparse and dense terms, which are utilized to during training learn the navigation behavior.

1) Goal Reaching: We reward goal reaching by providing a one-time reward of $r_g(s,a) = +0.75$ when the agent reaches the goal. Additionally, at every time step, we reward the robot's progress towards the goal. Our progress reward is given by,

$$r_p(s,a) = 0.25 * \frac{d_g^{t-1} - d_g^t}{d_g^0},$$

where d_g^t, d_g^{t-1} is the distance to goal at time step t and $t-1$ respectively. And d_g^0 denotes the initial distance to the goal. The progress reward is such that when the robot reaches the goal (i.e., $d_g^N = 0$) the summation of the progress reward over the time steps and the goal reward, ideally sums to 1. That is,

$$r_g(s,a) + \sum^{N} r_p(s,a) = 0.75 + 0.25 \cdot \sum^{N} \frac{d_g^{t-1} - d_g^t}{d_g^0} = 1.$$

2) Penalizing Collisions: In cases of collisions with either walls or other agents, we penalize the agent with $r_c = -0.50$. Furthermore, to discourage the agent from navigating too closely to obstacles, we impose a penalty of $r_{prox.} = -0.1 - d_{obs}$ if the distance to the nearest obstacle falls below a threshold of $0.2m$, otherwise $r_{prox.} = 0.0$.

3) Penalize Jerky Motion: The action reward penalizes large angular deviations between time steps. The action reward is given by, $r_a(s,a) = -0.005 * \Delta\theta^2$.

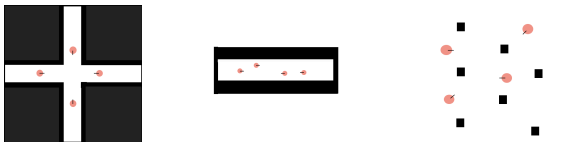
4) Penalize Excessive Communication: The communication reward limits the number of other robots an ego-robot communicates with. At each time step, the communication reward is given by, $r_c(s,a) = -0.0005 \cdot n_c$. The overall reward is given below.

$$R(s,a) = \mathcal{I}_{coll.} \cdot r_{coll.} + (1 - \mathcal{I}_{coll.}) \cdot (r_p + r_g + r_a + r_c + r_{prox}) \quad (6)$$

Where, I_{coll} is an indicator function that turn 1 when in collision and 0 otherwise.

V. Evaluations

In this section, we present our experimental setup and the evaluation of our method against other prior state-of-the-art methods.



(a) Intersection **(b)** Corridor **(c)** Random Obstacle

Fig. 2: Training environments

A. Training Setup

Our method was trained on an Intel Xeon 4208 CPU with 32 GB RAM and an Nvidia GeForce RTX 2080 Ti graphic card. We use TensorFlow and Python for the deep learning implementation. We build on the gym-collision avoidance and GA3C-CADRL package [12] to implement our method and to run our evaluations.

We use GA3C, an actor-critic method for training our network. We train the network in multiple stages, by gradually increasing the scenario complexity. Initially, the network undergoes training in a scenario involving 2 agents for the first $200k$ episodes. Subsequently, it proceeds to train until convergence with 4 agents. Finally, the network is trained with 8 agents until convergence. We use different static maps for our training process. The scenarios ranges from free space, intersection (Fig. 2-a), narrow passage (Fig. 2-b), and sparsely distributed obstacles (Fig. 2-c). Our real-world setup uses TurtleBots, and we use ROS topics to communicate between the agents. The robot uses AMCL package[1] for self-localization, and their observable states are communicated to other robots using ROS topics.

B. Evaluation Metrics

We compare our method against prior decentralized baselines including ORCA [18], CADRL [10], GA3C-CADRL [12] in the following evaluations. We evaluate performance using the following metrics:

- **Success Ratio (SR):** The ratio of robots that successfully reached their goal without collisions.
- **Collision Ratio (CR):** The ratio of robots that collided.
- **Timeout Ratio (TR):** Proportion of agents that neither collided nor reached their goal within 500 time steps.
- **Overall Time to Goal (T_g^{total}):** The time step at which all agents have successfully reached their goals without any collisions.
- **Agent Time to Goal (T_g^{agent}):** The average time required for an agent to reach its goal location.
- **Path Ratio:** The ratio between the length of an agent's actual path to its goal and the straight-line path length to the goal.
- **Normalized Communication Request (NCR):** The ratio of communication requests generated by the network for an agent compared to the maximum number of requests possible under a full communication policy (such as broadcast).

C. Evaluation Scenarios

We evaluate the proposed method against selected baseline approaches across various scenarios to assess their navigation performance.

[1]http://wiki.ros.org/amcl

Agents	Method	SR	CR	TR	T_g^{total}	T_g^{agent}	Path Factor (↓)
4	ORCA [18]	1.00	0.00	0.00	145.50	144.93	1.00
	CADRL [10]	1.00	0.00	0.00	157.90	156.65	1.14
	GA3C [12]	1.00	0.00	0.00	142.40	141.35	1.03
	PM*	1.00	0.00	0.00	150.8	147.9	1.07
8	ORCA [18]	1.00	0.00	0.00	177.6	176.8	1.01
	CADRL [10]	1.00	0.00	0.00	157.90	154.11	1.14
	GA3C [12]	1.00	0.00	0.00	143.70	142.37	1.04
	PM*	1.00	0.00	0.00	151.00	148.52	1.07
16	ORCA [18]	1.00	0.00	0.00	207.40	176.65	1.23
	CADRL [10]	1.00	0.00	0.00	164.90	152.08	1.19
	GA3C [12]	0.33	0.67	0.00	-	-	-
	PM*	1.00	0.00	0.00	166.00	158.81	1.10
32	ORCA [18]	0.71	0.19	0.00	260.5	189.63	1.54
	CADRL [10]	0.98	0.00	0.00	185.63	160.90	1.29
	GA3C [12]	0.00	1.00	0.00	-	-	-
	PM*	1.00	0.00	0.00	188	175.88	1.15
50	ORCA [18]	0.47	0.33	0.20	-	-	-
	CADRL [10]	0.94	0.06	0.00	220	172.52	1.49
	GA3C [12]	0.00	1.00	0.00	-	-	-
	PM*	0.99	0.01	0.00	215.2	189.96	1.19

TABLE II: Circle Scenario: The table presents a comparative analysis of our method against various baseline approaches across different navigation metrics. The complexity progressively increases as we increase the number of agents in the scenarios, causing congestion near the center of the circle. We observe that the proposed method (PM*) consistently results in the best path factor and success rate. Additionally, the time to goal in these scenarios is either the best or closely rivals the best results achieved.

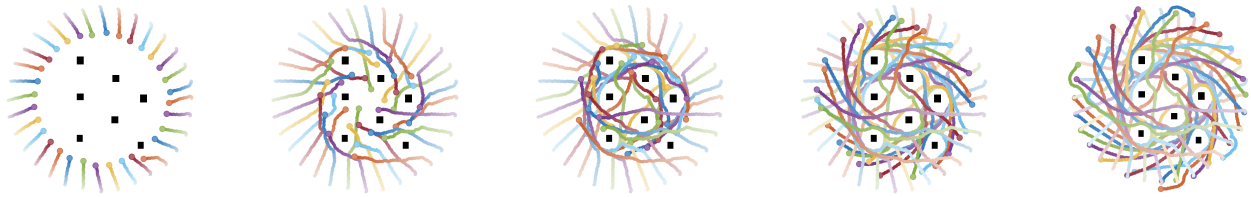
1) Circle Scenario: In this scenario, robots are evenly distributed along the circumference of a circle, each tasked with moving to its diagonally opposite position. To introduce diversity in start and goal configurations, we introduce a small noise to the (x, y) coordinates of both start and goal locations. For this evaluation, we consider between 4 and 50 agents and we tabulate the results in Table II. We observe that the proposed methods result in the best success rate across all the test cases. Moreover, in cases with 16 or more agents, it results in the best path factor, and total time to goal as well.

2) Formation Scenario: In this case, we have 16 robots arranged into a 4×4 grid. The formation task is such that the agent at $(\texttt{row}, \texttt{column})$ swaps its position with agent at $(\texttt{n-row+1}, \texttt{n-column+1})$. In table III, we tabulate the the evaluation metrics for robot radii varying between $0.4m$ to $0.6m$, thereby increasing the scenario density. We observe the proposed method results in the highest success rate.

3) Static Obstacles: We evaluate our method in a scenario with 32 robots moving in an environment with static obstacles. The robots leverage their local occupancy map information to guide their navigation around these obstacles. In figure 3, we illustrate the resulting trajectories in this scenario where our method navigates the robots around the static obstacles and other robots safely. In Figure 4, a real-world implementation of two robots operating in a static environment is presented.

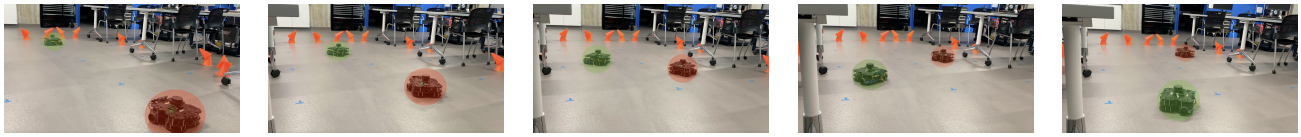
D. Selective Communication

To evaluate the efficacy of selective communication, we tabulate the likelihood of communication relative to the distance to a specific neighbor. For this analysis, we collected data points from multiple runs of the evaluation in the circle scenario. Particularly, the data points include the agent's distance from its neighbor and whether it requested to communicate. We then discretized the inter-agent distances to the nearest integer and use them as intervals. We ensure a



(a) $t = 5s$ **(b)** $t = 15s$ **(c)** $t = 25s$ **(d)** $t = 35s$ **(e)** $t = 45s$

Fig. 3: Snapshots of 32 agents navigating toward their respective goals within an environment with multiple static obstacles (depicted in black). The proposed method effectively guides each robot to its goal, avoiding collisions with both static obstacles and other agents.



(a) $t = 5s$ **(b)** $t = 15s$ **(c)** $t = 25s$ **(d)** $t = 35s$ **(e)** $t = 45s$

Fig. 4: Real-world Navigation: The scenario involves two robots navigating to interchange their positions amidst static obstacles in the form of table and traffic cones. The robots successfully navigates by avoid collisions with the static obstacles (table) and the other robot.

Radius	Method	SR	CR	TR
0.4	ORCA [18]	0.00	0.00	1.00
	CADRL [10]	0.38	0.56	0.06
	GA3C [12]	0.00	1.00	0.00
	PM*	**1.00**	**0.00**	**0.00**
0.5	ORCA [18]	0.00	0.00	1.00
	CADRL [10]	0.44	0.25	0.31
	GA3C [12]	0.00	1.00	0.00
	PM*	**1.00**	**0.00**	**0.00**
0.6	ORCA [18]	0.00	0.00	1.00
	CADRL [10]	0.43	0.31	0.25
	GA3C [12]	0.00	0.00	1.00
	PM*	**1.00**	**0.00**	**0.00**

TABLE III: Formation Scenario: The table presents the success ratio (SR), collision ratio (CR), and timeout ration (TR) for the formation scenario with 16 agents. As the agent radius increases, the scenario becomes more dense. ORCA results in a deadlock due to the highly symmetric nature of this scenario. The proposed Method (PM*) shows the best performance, resulting in $2\times$ better success rate than the second best. We highlight that the network was not trained in this scenario during the learning phase.

minimum of 1000 data points for each interval to maintain statistical significance. The table illustrates the fraction of instances within each distance bin where communication is requested, interpreted as the likelihood of communication. In Table IV, we observe a correlation between inter-agent distance and the decision to communicate. The likelihood is highest when agents are in close proximity, while still lower compared to broadcasting, where the likelihood is 1.

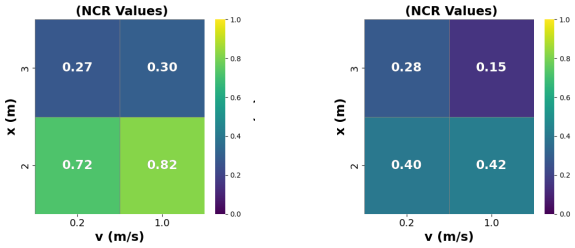
Additionally, we investigate the impact of inter-agent distance and velocity on the communication decision-making process. To gain deeper insights, we use a scenario involving two agents, each with a radius of 0.25 meters, and analyze two distinct cases:

- **Case 1:** Agent 1 begins at position $(0, 0)$ and navigates to a goal at $(0, 4)$, while Agent 2 starts at position $(x, 0)$ and navigates to $(x, 4)$. Consequently, the agents move nearly parallelly, maintaining an inter-agent distance of approximately $x - 0.5m$.
- **Case 2:** Agent 1 maintains the same initial and goal positions as in Case 1, while Agent 2 starts at position $(x, 4)$ and moves to $(x, 0)$, resulting in the agents moving in opposite directions.

Considering two values for x ($2m$ and $3m$) and two values

Inter-Agent Distance	Communication Likelihood
1	0.67
2	0.50
3	0.23
4	0.25
5	0.22
6	0.23
7	0.17

TABLE IV: We tabulate the communication likelihood of our method with respect to inter-agent distance with the neighbor. We observe our network communicates more when inter-agent distance is low. While for inter-agent distance more than $2m$, it averages around $\sim 20\%$ of the time.



(a) Case 1 **(b)** Case 2

Fig. 5: Correlation Analysis: We consider two scenarios, one where two agents move in parallel in the same direction and another where they move towards each other. Our observations indicate that with a decrease in inter-agent distance, there is a corresponding increase in the likelihood of communication in the network.

for agent velocity ($1m/s$ and $0.2m/s$), we compare the normalized communication requests generated in our method for these cases. The results are illustrated as a heatmap in Fig. 5. We observe that the inter-agent distance largely affects the communication decision. The normalized communication request (NCR) is higher when the agents are in close proximity. For instance, in Case 1, the NCR drop by more than half when x is increased from $2m$ to $3m$. Similarly, in Case 1, when the agent velocity in reduced from $1m/s$ to $0.2m/s$, the NCR is reduced by $\sim 10\%$. In Case 2, we observe a similar trend with respect to x, but when the velocity is decreased the NCR is seen to increase.

VI. Conclusion and Future Work

We presented a deep reinforcement learning (RL) method for dense multi-robot navigation in complex scenarios. Our

method learns to navigate effectively and to communicate intelligently, determining what information to convey, when and whom to communicate with. Our approach is shown to outperform existing state-of-the-art methods, particularly in navigating through complex scenarios. Furthermore, we conducted experiments using a small group of ground robots to validate our method's efficacy in real-world settings. Similar to other learning-based approaches, a limitation of our work is that we cannot provide guarantees regarding safety or navigation performance. As a direction for future research, we aim to explore the impact of the communication vector's size on the overall navigation performance. Additionally, we are interested in understanding the semantics of the learned communicated messages.

REFERENCES

[1] X. An, C. Wu, Y. Lin, M. Lin, T. Yoshinaga, and Y. Ji, "Multi-robot systems and cooperative object transport: Communications, platforms, and challenges," *IEEE Open Journal of the Computer Society*, vol. 4, pp. 23–36, 2023.

[2] S. Tang and V. Kumar, "A complete algorithm for generating safe trajectories for multi-robot teams," in *Robotics Research*. Springer, 2018, pp. 599–616.

[3] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[4] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998. [Online]. Available: https://doi.org/10.1177/027836499801700706

[5] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, online collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[6] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, "Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 236–243.

[7] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete rrt for exploration of implicit roadmaps in multi-robot motion planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 501–513, 2016.

[8] M. Goldenberg, A. Felner, R. Stern, G. Sharon, N. Sturtevant, R. C. Holte, and J. Schaeffer, "Enhanced partial expansion a," *Journal of Artificial Intelligence Research*, vol. 50, pp. 141–187, 2014.

[9] H. Murakami, C. Feliciani, Y. Nishiyama, and K. Nishinari, "Mutual anticipation can contribute to self-organization in human crowds," *Science Advances*, vol. 7, no. 12, p. eabe7758, 2021. [Online]. Available: https://www.science.org/doi/abs/10.1126/sciadv.abe7758

[10] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 285–292, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:8342451

[11] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6252–6259.

[12] M. Everett, Y. F. Chen, and J. P. How, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 10 357–10 377, 2021.

[13] Z. Ma, Y. Luo, and H. Ma, "Distributed heuristic multi-agent path finding with communication," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8699–8705.

[14] Serra-Gómez, H. Zhu, B. Brito, W. Böhmer, and J. Alonso-Mora, "Learning scalable and efficient communication policies for multi-robot collision avoidance," *Autonomous Robots*, vol. 47, pp. 1–23, 08 2023.

[15] S. H. Arul, A. S. Bedi, and D. Manocha, "Dmca: Dense multi-agent navigation using attention and communication," 2022.

[16] S. Paris, J. Pettre, and S. Donikian, "Pedestrian reactive navigation for crowd simulation: a predictive approach abstract," *Comput. Graph. Forum*, vol. 26, pp. 665–674, 09 2007.

[17] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1928–1935.

[18] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*, C. Pradalier, R. Siegwart, and G. Hirzinger, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 3–19.

[19] S. H. Arul and D. Manocha, "V-rvo: Decentralized multi-agent collision avoidance using voronoi diagrams and reciprocal velocity obstacles," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 8097–8104.

[20] Y. Yongjie and Z. Yan, "Collision avoidance planning in multi-robot based on improved artificial potential field and rules," in *2008 IEEE International Conference on Robotics and Biomimetics*, 2009, pp. 1026–1031.

[21] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[22] S. H. Arul and D. Manocha, "Dcad: Decentralized collision avoidance with dynamics constraints for agile quadrotor swarms," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1191–1198, 2020.

[23] R. Chandra, V. Zinage, E. Bakolas, J. Biswas, and P. Stone, "Decentralized multi-robot social navigation in constrained environments via game-theoretic control barrier functions," *arXiv preprint arXiv:2308.10966*, 2023.

[24] G. Berseth, M. Kapadia, B. Haworth, and P. Faloutsos, "Steerfit: automated parameter fitting for steering algorithms," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '14. Goslar, DEU: Eurographics Association, 2015, p. 113–122.

[25] I. Karamouzas, B. Skinner, and S. J. Guy, "Universal power law governing pedestrian interactions," *Phys. Rev. Lett.*, vol. 113, p. 238701, Dec 2014. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.113.238701

[26] B. Davis, I. Karamouzas, and S. Guy, "Nh-ttc: A gradient-based framework for generalized anticipatory collision avoidance," 07 2020.

[27] T. Balch and R. C. Arkin, "Communication in reactive multiagent robotic systems," *Autonomous robots*, vol. 1, pp. 27–52, 1994.

[28] J. Godoy, I. Karamouzas, S. J. Guy, and M. Gini, "Implicit coordination in crowded multi-agent navigation," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, p. 2487–2493.

[29] G. Best, M. Forrai, R. R. Mettu, and R. Fitch, "Planning-aware communication for decentralised multi-robot coordination," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1050–1057.

[30] B. Boardman, T. Harden, and S. Martínez, "Multi-agent motion planning with sporadic communications for collision avoidance," *IFAC Journal of Systems and Control*, vol. 15, p. 100126, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468601820300298

[31] D. Hildreth and S. J. Guy, "Coordinating multi-agent navigation by learning communication," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 2, no. 2, jul 2019. [Online]. Available: https://doi.org/10.1145/3340261

[32] A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau, "Tarmac: Targeted multi-agent communication," in *International Conference on machine learning*. PMLR, 2019, pp. 1538–1546.

[33] Z. Ma, Y. Luo, and J. Pan, "Learning selective communication for multi-agent path finding," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1455–1462, 2021.

[34] Z. Ding, T. Huang, and Z. Lu, "Learning individually inferred communication for multi-agent cooperation," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 22 069–22 079.

[35] Y. Zhai, B. Ding, X. Liu, H. Jia, Y. Zhao, and J. Luo, "Decentralized multi-robot collision avoidance in complex scenarios with selective communication," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8379–8386, 2021.

[36] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

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