

Learning scalable and efficient communication policies for multi-robot collision avoidance

Álvaro Serra-Gómez^{1*}, Hai Zhu¹, Bruno Brito¹, Wendelin Böhmer² and Javier Alonso-Mora¹

¹Department of Cognitive Robotics, Delft University of Technology, Mekelweg 2, Delft, 2628 CD, Netherlands.

²Algorithmics group, Delft University of Technology, Van Mourik Broekmanweg 6, Delft, 2628 XE, The Netherlands.

*Corresponding author(s). E-mail(s): a.serragomez@tudelft.nl;

Contributing authors: h.zhu@tudelft.nl; bruno.debrito@tudelft.nl; j.w.bohmer@tudelft.nl; j.alonsomora@tudelft.nl;

Abstract

Decentralized multi-robot systems typically perform coordinated motion planning by constantly broadcasting their intentions to avoid collisions. However, the risk of collision between robots varies as they move and communication may not always be needed. This paper presents an efficient communication method that addresses the problem of “when” and “with whom” to communicate in multi-robot collision avoidance scenarios. In this approach, each robot learns to reason about other robots’ states and considers the risk of future collisions before asking for the trajectory plans of other robots. We introduce a new neural architecture for the learned communication policy which allows our method to be scalable. We evaluate and verify the proposed communication strategy in simulation with up to twelve quadrotors, and present results on the zero-shot generalization/robustness capabilities of the policy in different scenarios. We demonstrate that our policy (learned in a simulated environment) can be successfully transferred to real robots.

Keywords: collision avoidance, multi-robot communication, multi-agent reinforcement learning, aerial robots, multi-robot systems.

1 Introduction

Being able to account for the future trajectory of other robots is of utmost importance for safe navigation in environments shared with other robots. Centralized systems achieve this objective by using a central station to manage all of the robots’ information and plans but they are difficult to scale up to large teams. Instead, decentralized systems can scale up by relying on

each robot’s on-board computation capabilities. Some decentralized solutions have each robot estimate other robots’ behaviors or future trajectories through trained parametric functions, e.g. neural networks (Zhu, Claramunt, Brito, & Alonso-Mora, 2021). However, these solutions are generally computationally expensive and may have inaccuracies stemming from the lack of information on the other robot’s goals and local observations. Instead, direct communication of each robot’s trajectory

intentions may allow to obtain accurate predictions with less computational effort. Common communication policies broadcast to all robots, employing distance-based heuristics to communicate trajectory plans. However, much of this information becomes redundant or unnecessary when robot motions do not present a threat to others, e.g. when they are far from each other, or may, at worst, harm the multi-robot system’s performance (Talamali, Saha, Marshall, & Reina, 2021). Besides, designing a set of rules to communicate efficiently may be difficult as it would require to estimate a priori the future value of communicating with other robots, which also depends on their motion planner and dynamics. Since there is no clear intuition on how to hand-engineer an adequate trade-off between communication efficiency and safety, in this work, we focus on the following two issues: **a)** providing a solution to the problem of when and with whom to communicate that can scale up to large teams of robots, and **b)** how to couple this communication policy with existing motion planning methods.

We propose an efficient communication policy method combined with an optimal control motion planner for multi-robot collision avoidance that can handle large multi-robot systems with varying number of robots. The approach leverages the strengths of learning methods for decision-making and nonlinear receding horizon control, or Non-Linear Model Predictive Control (NMPC) for multi-robot motion planning. In particular, we use Multi-Agent Reinforcement Learning (MARL) to learn the robots’ communication policies. For every robot and time instance, the policy selects a set of other robots (Wheeler, Bharathi, & Gil, 2019) and requests their trajectory plans. Non-selected robots are assumed to follow their last communicated which trajectory extended assuming constant velocity. This last communicated trajectory is exploited as long as they remain within a tolerance distance. Otherwise, robots that do not follow their last communicated trajectory are assumed to follow a constant velocity trajectory. Then, we formulate a nonlinear optimization problem to generate a safe trajectory. The planned trajectory takes into account the requested and estimated trajectories represented as constraints in the receding horizon framework.

The main contributions of this work are:

- A combined communication policy and trajectory planning method for micro-aerial vehicles (MAVs), that utilizes the strengths of nonlinear model predictive control (NMPC) to plan safe trajectories, and multi-agent reinforcement learning (MARL) to learn an efficient communication policy.
- An on-line communication policy that uses MARL to learn (off-line) *when* and with *whom* it is useful to communicate, decreasing the amount of communication while still achieving safe navigation and coordination among robots.
- We introduce a new neural architecture for the learned communication policy that scales to large and varying number of robots while still providing safe navigation in a variety of situations.
- We demonstrate that the communication policy, which is trained in a simulator, works equally well in physical MAVs.

We evaluate our method with team of varying number of quadrotors in simulated scenarios requiring different levels of interaction for safe navigation and compare it with four other heuristic based methods for communication. We then test the robustness of our method under different levels of observation noise. Finally, we show that our method presents zero-shot generalization properties when tested in scenarios with more robots than during training while still maintaining safety online.

In an earlier conference version of this work (Serra-Gómez, Brito, Zhu, Chung, & Alonso-Mora, 2020), an early version of the framework to learn a communication policy and its combination with a local motion planner was introduced for a fixed number of robots. In this paper, we extend the approach with a new neural architecture and refine the training procedure of the communication policy to render the final navigation policy safer in interaction-rich situations, more robust to sampled training scenarios and scalable to robot teams of varying number of robots. We show that our learning method enables the emergence of more efficient and intuitive communication behaviours than before, while maintaining a performance similar to that of broadcasting policies with regards to safe navigation.

2 Related Work

2.1 Communication in Collision Avoidance

We focus our work on online local motion planning for multi-robot systems (also referred as multi-robot collision avoidance), which has been actively studied over the past years. Traditional reactive controller-level approaches include the optimal reciprocal collision avoidance (ORCA) method (Van Den Berg, Guy, Lin, & Manocha, 2011), the artificial potential field (APF) based method (Yongjie & Yan, 2009), the buffered Voronoi cell (BVC) approach (Zhou, Wang, Bandyopadhyay, & Schwager, 2017; Zhu & Alonso-Mora, 2019a), and control barrier functions (CBF) (L. Wang, Ames, & Egerstedt, 2017). These methods are fully decentralized and each robot only needs to know other robots' current state, which can be measured by the robot via its onboard sensors. Hence, communication among robots is not necessary. However, these reactive methods are inefficient since they typically plan one time step ahead. This can result in overly conservative policies that are more vulnerable to deadlocks than predictive collision avoidance methods. These issues can be overcome by using a model predictive control (MPC) framework for collision-free trajectory generation that accounts for the plans of other robots (Zhu & Alonso-Mora, 2019b).

For each robot to solve a local trajectory optimization problem in the MPC framework, it needs to know the future trajectories of other robots. One approach is to let each robot communicate its planned trajectory with every other robot in the team. Hence, robots can then update their own trajectories to be collision free with other robots' trajectory plans, as in these distributed MPC works (Luis, Vukosavljev, & Schoellig, 2020; Zhu & Alonso-Mora, 2019b). Another approach is to let each robot predict other robots' future motions based on its own observations. For instance, Kamel, Alonso-Mora, Siegart, and Nieto (2017) employs a constant velocity model when predicting other robots' future trajectories. In that case, communication among robots is not required. However, such a prediction can be inaccurate and may lead to unsafe trajectory planning, in particular when the robots are moving at a high speed (Zhu & Alonso-Mora, 2019b). In this paper,

we aim to develop a communication policy, which determines “when” and “with whom” a robot should communicate with another robot in the system, to reduce the amount of communication while still keeping a high-level of safety.

2.2 Communication Scheduling

A lot of works tend to formulate the problem of efficient communication in a receding horizon fashion. Some methods formulate the problem as a decentralized version of a Markov Decision Process (Dec-MDP) (Roth, Simmons, & Veloso, 2005) or Partially Observable MDP (Dec-POMDP) (Becker, Carlin, Lesser, & Zilberstein, 2009) and try to optimize a value function in which communications are penalized. Others, such as Kassir, Fitch, and Sukkarieh (2016), choose to formulate a constrained optimization problem where communications must be directly minimized while still guaranteeing data flow throughout the network. These approaches assume access to an analytical model or require the design of a hand-engineered utility function to estimate the future effects of communication, which might not be available or intuitive to do, respectively. Recent work (Best, Forrai, Mettu, & Fitch, 2018) manages to tackle this problem by triggering communication whenever uncertainty over another agent's actions exceeds a threshold. Ultimately, however, receding horizon methods are limited by their prediction horizon and the need for hand-engineered evaluation heuristics, which can unintentionally bias the resulting communication processes. In this work, we use reinforcement learning methods to learn the communication policy. Through learning from experience, this family of methods has the potential to discover more general policies without the need for fine-tuning hand-engineered heuristic functions..

2.3 Learning Methods for Coordination

One major challenge in Multi-Agent Reinforcement Learning (MARL) is the non-stationarity of multi-agent environments. This problem is caused by having multiple agents that learn and change their policy every learning iteration, which may result in the learning process being unstable. In order to mitigate this challenge, recent

works on MARL (Foerster, Farquhar, Afouras, Nardelli, & Whiteson, 2018; Iqbal & Sha, 2019; Lowe et al., 2017; Rashid et al., 2018; Son, Kim, Kang, Hostallero, & Yi, 2019; Sunehag et al., 2018) perform centralized training and decentralized execution. This paradigm has been applied in the field of non-communicating multi-robot collision avoidance tasks (Everett, Chen, & How, 2018; Everett, Chen, & P. How, 2019) to learn an end-to-end navigation policy. Yet, these methods typically do not offer solid theoretical guarantees for collision avoidance. Instead we aim to learn the communication policy while leveraging already existing well-performing motion planners, e.g. (Zhu & Alonso-Mora, 2019b).

Regarding tasks that require communication, several works have been published recently. Many of them focus on learning what content should be shared among agents, be it in the form of explicit messages (Li, Gama, Ribeiro, & Prorok, 2020), a composition of binary signals (Foerster, Assael, de Freitas, & Whiteson, 2016) and predefined symbols (Mordatch & Abbeel, 2018), policy hidden layers (Sukhbaatar, Szlam, & Fergus, 2016), or by directly sharing parameters among agents (Gupta, Egorov, & Kochenderfer, 2017). The most relevant to our work additionally focus on learning, in a scalable way, policies that are able to appropriately choose *when* and with *whom* to communicate or cooperate in collision avoidance tasks. Jiang and Lu (2018) assign roles to every agent, making some of them in charge of organizing a common communication channel with their neighbours. However, regions where there is no agent with such a role are left without coordination capabilities. Instead, Das et al. (2019), Li, Lin, Liu, and Prorok (2020) and Zhai et al. (2021) present end-to-end MARL algorithms that design an attention module to assign and weight the importance of the messages received from other agents. While previous methods use dense attention mechanisms, Sun, Shen, and How (2020) proposes an adaptive sparsity-inducing activation function to enable learning a sparse communication graph. Along these lines, Ding, Huang, and Lu (2020) learn to choose whom to communicate with and evaluate the received messages to choose an action.

Similarly, the method we present in this paper can also be considered as an attention module targeting other agents. However, we set our

communications to be unilateral to promote asymmetrical behaviour. Additionally, we decouple the problem of communication and motion planning, allowing the combination of our method with existing and well-tested solutions for motion planning in collision avoidance tasks.

3 Preliminaries

In this paper, we address the problem of deciding when and with whom to communicate during a multi-robot collision avoidance task. Though the proposed formulation is intended to be general, we are inspired by the results obtained in Zhu and Alonso-Mora (2019b), which show how in a collision-avoidance scenario, methods that incorporate communication have a clear advantage over those that do not. We approach the information-sharing process as a MARL problem where the robots must learn to request information effectively. In this section, we set the context for our targeted communication process by formulating the problem of multi-robot collision avoidance. We provide an overview of the Non-Linear Model Predictive Control method used for motion control, as well as our MARL framework, introducing relevant notations for this work.

3.1 Multi-Robot Collision Avoidance

Consider a team of n robots moving in a shared workspace $\mathcal{W} \subseteq \mathbb{R}^3$, where each robot $i \in \mathcal{I} = \{1, 2, \dots, n\} \subset \mathbb{N}$ is modeled as an enclosing sphere with radius r . The dynamics of each robot $i \in \mathcal{I}$ are described by a discrete-time equation as follows,

$$\mathbf{x}_i^{t+1} = \mathbf{f}(\mathbf{x}_i^t, \mathbf{u}_i^t), \quad \mathbf{x}_i^0 = \mathbf{x}_i(0), \quad (1)$$

where $\mathbf{x}_i^t \in \mathcal{X} \subset \mathbb{R}^{n_x}$ denotes the state of the robot with dimension n_x , typically including its position $\mathbf{p}_i^t \in \mathbb{R}^3$ and velocity $\mathbf{v}_i^t \in \mathcal{R}^3$ (amongst others, see section 4.1.1), and $\mathbf{u}_i^t \in \mathcal{U} \subset \mathbb{R}^{n_u}$ the control inputs with dimension n_u . The function f is the model of the robot and is detailed in Appendix A. The super-script \cdot^t indicates the time step t . \mathcal{X} and \mathcal{U} are the admissible state space and control space respectively. $\mathbf{x}_i(0)$ is the initial state of robot i . Any pair of robots i and j from the group are mutually collision-free if

$\|\mathbf{p}_i^t - \mathbf{p}_j^t\| \geq 2r, \forall i \neq j \in \mathcal{I}, \forall t = 0, 1, \dots$. Each robot has a given goal location \mathbf{g}_i , which generally comes from some high-level path planner or is specified by some user.

Robots in the team are allowed to communicate. Communication is assumed to be ideal, e.g. robots can communicate with each other perfectly and instantaneously. We also assume a point-to-point network topology. The implementation is viable provided that communication protocols with low energy consumption, such as a mesh network where links are established using Bluetooth LE, are utilized. Under this communication protocol, point-to-point communication topologies generally use less bandwidth than broadcasting topologies and scale better with the number of robots as they allow for redundant or unnecessary communication to be avoided. Robots can associate other robots with the messages they send since, in practice, each sender could add its ID to the message or the receiving robot could infer it using the first position of the received trajectory.

The objective of multi-robot collision avoidance is to compute a local motion \mathbf{u}_i^t for each robot in the group, that respects its dynamics constraints, makes progress towards its goal location \mathbf{g}_i and is collision-free with other robots in the team for a time horizon $\tau = N\Delta t$, where Δt is the sampling time and N is the number of discrete steps.

3.2 Distributed Model Predictive Control

The key idea of using distributed model predictive control to solve the multi-robot collision avoidance problem is to formulate it as a receding horizon constrained optimization problem. For each robot $i \in \mathcal{I}$, the discrete-time constrained optimization formulation is

$$\begin{aligned} \min_{\mathbf{x}_i^{0:N}, \mathbf{u}_i^{0:N-1}} & \sum_{k=0}^{N-1} J_i^t(\mathbf{x}_i^k, \mathbf{u}_i^k) + J_i^N(\mathbf{x}_i^N, \mathbf{g}_i) \\ \text{s.t.} & \mathbf{x}_i^0 = \mathbf{x}_i(0), \\ & \mathbf{x}_i^{k+1} = \mathbf{f}(\mathbf{x}_i^k, \mathbf{u}_i^k), \\ & \|\mathbf{p}_i^k - \mathbf{p}_j^k\| \geq 2r, \\ & \mathbf{u}_i^{k-1} \in \mathcal{U}, \quad \mathbf{x}_i^k \in \mathcal{X}, \\ & \forall j \neq i \in \mathcal{I}; \quad \forall k \in \{0, 1, \dots, N\}. \end{aligned} \quad (2)$$

Where $J_i^k(\mathbf{x}_i^k, \mathbf{u}_i^k)$ and $J_i^N(\mathbf{x}_i^N, \mathbf{g}_i)$ are the stage and terminal costs, respectively (Zhu & Alonso-Mora, 2019b) (defined on Appendix B). At each time step, each robot in the team solves online the constrained optimization problem (2) and then executes the first step control inputs, in a receding-horizon fashion. In this paper, the generated future plans of robot i are also called robot i 's (future) trajectory intentions.

3.3 Problem Formulation

For each robot to solve problem (2), it has to know the future trajectories of other robots in the team. Aside from the particular case of prioritized sequential motion planning schedules, obtaining the exact information on future trajectories beforehand is generally not feasible. Thus, other robots' future positions can be approximated either by estimating their predictions (Zhu et al., 2021) or by requiring them to communicate their trajectory intentions, computed during the previous time step (Zhu & Alonso-Mora, 2019b).

At time t , let $\hat{\mathcal{T}}_{j|i}^t = \{\mathbf{p}_j^{t+1:t+N} \mid \text{predicted at time } t\}$ be the N -time horizon trajectory of robot $j \in \mathcal{I}, j \neq i$ that robot i assumes and uses in solving the problem (2), where the hat $\hat{\cdot}$ indicates that it is what robot i knows about the other agent's trajectory. Further denote by $\mathcal{T}_i^t = \{\mathbf{p}_i^{t:t+N} \mid \text{predicted at time } t\}$ the trajectory for robot i planned at time t . As mentioned, there are two ways for robot i to approximate the future trajectory of robot j , namely $\hat{\mathcal{T}}_{j|i}^t$:

- **Without communication:** robot i predicts another robot's future trajectory based on their current states, that is

$$\hat{\mathcal{T}}_{j|i}^t = \text{prediction}(\mathbf{x}_j^t), \quad \forall j \neq i \in \mathcal{I}. \quad (3)$$

In Kamel et al. (2017), each robot was considered to follow constant velocity model for the prediction. However, this approach ignores the previously communicated information on future trajectory intentions from other robots, even when they could potentially hold more information on other robots' future positions than constant velocity estimates. Our prediction model uses the last communicated trajectory plans and expands it by assuming constant velocity. If robot j strays past a predetermined

distance from its last communicated trajectory intentions, then robot j is estimated to follow a constant velocity model from its current position (section 4.4).

- **Communication request:** Robot i can request other robots j in the team to communicate their planned trajectories at each time step, that is:

$$\hat{\mathcal{T}}_{j|i}^t(0 : N-1) = \mathcal{T}_j^{t-1}(1 : N), \quad \forall j \neq i \in \mathcal{I}. \quad (4)$$

where $\hat{\mathcal{T}}_{j|i}^t(a : b)$, with $a \leq b$, represents the subsequence of $\hat{\mathcal{T}}_{j|i}^t$ that goes from the a^{th} to the b^{th} -indexed element (inclusive). At time t , the last position of the communicated path of robot j : $\hat{\mathcal{T}}_{j|i}^t(N) = \hat{\mathbf{p}}_{j|i}^N$, cannot be communicated as it is beyond the N -time horizon at time step $t-1$. Therefore it is estimated by assuming constant velocity of robot j : $\hat{\mathbf{p}}_{j|i}^N = \hat{\mathbf{p}}_{j|i}^{N-1} + (\hat{\mathbf{p}}_{j|i}^{N-1} - \hat{\mathbf{p}}_{j|i}^{N-2})$.

Both of the two methods have their advantages and disadvantages. Fully communicating methods allow more accurate predictions and achieve safe collision avoidance as long as a feasible solution is found, but they require a large amount of communication among robots. If there is no communication, the robot may plan an unsafe trajectory if its prediction on other robots' trajectories deviates from their real ones or an overly conservative trajectory to avoid collisions.

Motivated by these facts, this paper aims to solve the problem of “with whom to communicate” for each robot in the team for collision avoidance. More precisely, at each time step, each robot i decides whether or not to request a trajectory intention from every other robot j . If robot i decides to request robot j , robot j should communicate its planned trajectory to robot i . If robot i decides not to request robot j , it predicts robot j 's future trajectory based on its last communicated trajectory intention and the observed current state of robot j .

Denote by $\pi_i^t = \{c_{j|i}^t \mid \forall j \neq i\}$ the communication vector of robot i at time t , in which $c_{j|i}^t = 1$ indicates that robot i requires a communicated trajectory from robot j . Otherwise $c_{j|i}^t = 0$. Note that $c_{i|i}^t = 0$ since the robot does not need to communicate with itself. Let $\pi^t = \{\pi_1^t; \dots; \pi_n^t\}$ be

the communication matrix of the multi-robot system at time t in an episode of length T_e . We define the communication cost of the system to be

$$C(\pi^t) = \frac{1}{N_c(n)} \sum_i^n \sum_j^n c_{j|i}^t. \quad (5)$$

Where $N_c(n) = n(n-1)/T_e$ is a normalization factor depending on the number of agents and the length of the episode, that represents the maximum amount of communications that can happen within a system of n robots across T_e timesteps. The objective of this paper is to find a policy for each robot i ,

$$\pi_i^t = \pi_i(\mathbf{x}_1^t, \mathbf{x}_2^t, \dots, \mathbf{x}_n^t) = \{c_{j|i}^t \mid \forall j \neq i\} \quad (6)$$

that minimizes $C(\pi^t)$ while ensuring that the robots are collision-free with each other in the system.

4 Method

An overview of the proposed method is given in Figure 1. It consists of two components: a learned communication policy, which we introduce as *WW2C*, that decides with whom to communicate, and a NMPC planner.

Every time step, based on its partial observation of the current joint state z_i^t , every robot targets a set of other robots π_i^t and requests their intended trajectory plans $\hat{\mathcal{T}}_{j|i}^t = \mathcal{T}_j^{t-1}$ according to a learnt parametric policy $\pi_{i,\theta_i}(z_i^t)$. Those robots not targeted are estimated to follow a previously communicated trajectory extended assuming constant velocity or, in case it is no longer useful, a constant velocity model $\hat{\mathcal{T}}_{j|i}^t = \text{prediction}(\mathbf{x}_j^t)$ as described in section 3.3.

A receding horizon optimization is then employed to plan the future intended trajectory \mathcal{T}_i^t for robot i . To guarantee the safety of such a trajectory, the resulting path is constrained to not intersect with $\hat{\mathcal{T}}_{j|i}^t$ for any $j \neq i$. The first action input from the computed plan is applied and a new observation is gathered. Along this work we assume that robots plan and execute actions in a synchronized fashion. While this assumption is necessary for learning the communication policy, it can be alleviated during test time since

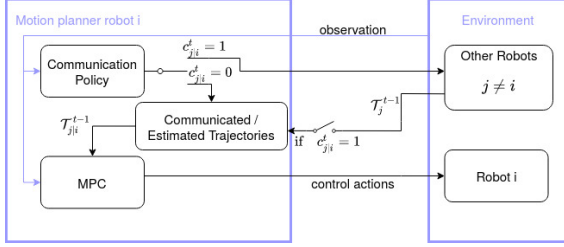


Fig. 1: Schema of the proposed method for efficient communication. $\pi_i^t(z_i)$ is the communication policy dependent on the observation z_i . \mathcal{T}_j^{t-1} is the trajectory plan of robot j at the previous time step. And $\hat{\mathcal{T}}_{j|i}^t$ is the combination of obtained and estimated trajectories of the other robots.

robots employ previously received or estimated trajectories (Zhu et al., 2021).

4.1 Reinforcement Learning Setup

We formulate a multi-robot reinforcement learning problem to compute an efficient communication policy. By considering the MPC based motion planner as part of the transition function, this problem can be transformed into a decentralized POMDP (Bernstein, Givan, Immerman, & Zilberstein, 2002). The decentralized POMDP is composed of six components, including state space, action space, observation space, reward function, transition model and observation model.

4.1.1 State space \mathcal{X}

For every robot i , $\mathbf{x}_i \in \mathcal{X}$ must account for the current physical state, its goal position, its sequence of intended future positions, computed during the previous time step by the motion planner, and its knowledge on other robots' future trajectory plans. Therefore, the state at time t can be defined as:

$$\mathbf{x}_i^t := [\mathbf{g}_i, \mathbf{p}_i^t, \mathbf{v}_i^t, \mathcal{T}_i^{t-1}, \hat{\mathcal{T}}_{-i|i}^{t-1}], \quad (7)$$

$$\mathbf{X}^t := \{\mathbf{x}_1^t, \mathbf{x}_2^t, \dots, \mathbf{x}_n^t\}, \quad (8)$$

where $\mathbf{g}_i, \mathbf{p}_i^t, \mathbf{v}_i^t \in \mathbb{R}^3$ are the goal, position and velocities of robot i at time t , and $\hat{\mathcal{T}}_{-i|i}^{t-1} = \{\hat{\mathcal{T}}_{j|i}^{t-1} \mid \forall j \neq i\}$. Then, \mathbf{X}^t is the joint state of the whole multi-robot system. Following a similar formulation as Everett et al. (2019), robot i only has access to the information of its own state \mathbf{x}_i^t and

the terms from other robots j that can be estimated through its sensors, such as their positions and velocities.

4.1.2 Observation space \mathcal{Z}

We assume all robots are within sensor range (e.g. camera, lidar, ...) of each other and can always estimate the relative positions and velocities of all other robots. Each robot also knows the relative position of its own goal from a mission planner. For robot i , partial observations on the joint state at time t are:

$$\mathbf{z}_i^t = [\mathbf{v}_i^t, \mathbf{p}_{i,g}^t, \{\mathbf{d}_{j|i}^t\}_{j \in I \setminus i}, \{\mathbf{p}_{j|i}^t\}_{j \in I \setminus i}, \{\mathbf{v}_{j|i}^t\}_{j \in I \setminus i}], \quad (9)$$

where $d_{j|i}^t, \mathbf{p}_{j|i}^t$ and $\mathbf{v}_{j|i}^t$ are the relative distances, positions and velocities of the other robots with respect to the i^{th} robot, and $\mathbf{p}_{i,g}^t$ is the relative position of robot i 's goal. The joint observation from all robots is denoted by $\mathbf{z}^t = \{\mathbf{z}_1^t, \dots, \mathbf{z}_n^t\} \in \mathcal{Z}$

4.1.3 Action space $\mathcal{A} = \times_{i \in \mathcal{I}} \mathcal{A}_i$

As it has already been introduced in section 3.3, we denote by $\pi_i^t = \{c_{j|i}^t \mid \forall j \neq i\}$ the communication vector of robot i at time t . Note we have dropped the i^{th} element as the robot cannot communicate with itself. Therefore the action space for robot i is:

$$\mathcal{A}_i = \{0, 1\}^{n-1}$$

Note that the dimensionality of the action space depends on the number of agents. This matter will be further addressed later in 4.2.

4.1.4 Reward $R_i(\mathbf{x}^t, \pi^t)$

The reward function is chosen based on the multiple behaviors we want to achieve. It aims for the learned communication policy to communicate as little as possible while allowing each robot in the team to reach its goal and avoid collisions. The reward value $R(\mathbf{x}^t, \pi^t)$ is the immediate reward that every robot i gets at a state $\mathbf{x} \in \mathcal{X}$ after applying the communication vector π_i^t .

Global rewards are often used in multi-agent systems in order to capture coordinating coupled behaviors. However, this often leads to multi-agent credit assignment problems during training (Rahmattalabi, Chung, Colby, & Tumer, 2016). In this

work we attempt to capture coupled behaviors by employing the same reward function conditioned to each robot’s state individually. Since all agents share the same architecture and policy parameters, this allows to quickly learn to properly punish pairwise interactions such as collisions and communications as all samples from all robots can be used in the same way to compute the gradients and update the parameters of the communication policy. Also, even though reward signals are individual, optimising the same set of parameters for all agents at the same time allows to account for coordinating behaviours when there are more than two agents crossing paths. The reward function is composed of the following weighted combination of terms:

$$R_i(\mathbf{x}^t, \pi_i^t) = w_g R_{g,i}(\mathbf{x}^t) + w_{coll} R_{coll,i}(\mathbf{x}^t) + w_c R_{c,i}(\pi^t) \quad (10)$$

where w_g , w_{coll} , w_c are the weights for each term. Each reward term is defined:

$$R_{g,i}(\mathbf{x}^t) = \begin{cases} r_g & \|\mathbf{p}_{i,g}^t - \mathbf{p}_i^t\| \leq r_i \\ 0 & \text{otherwise} \end{cases}$$

with $r_g > 0$ is a tuned reward given at the end of the episode if robot i is within its goal, r_i is the radius of the smallest sphere containing the robot. This reward gives an incentive to learn communication patterns that stir the robot toward its own goal.

$$R_{coll,i}(\mathbf{x}^t) = \begin{cases} -r_{coll} & \forall j \in \mathcal{I}, i \neq j, \\ & \|\mathbf{p}_i^t - \mathbf{p}_j^t\| \leq r_i + r_j \\ 0 & \text{otherwise} \end{cases}$$

where $r_{coll} > 0$ is a tuned penalty term for the collision between any two robots.

Finally the local penalization term for path plan requests is similar to the global version introduced before in section 3.3 and has the form:

$$R_{c,i}(\pi^t) = -C_i(\pi^t) = -\frac{1}{N_i(n)} \sum_{j \neq i}^n c_{j|i}^t.$$

Where $N_i(n)$ is a local normalization term depending on the number of agents.

4.1.5 Observation model

$$\mathcal{O}(\mathbf{z}^{t+1}, \mathbf{x}^{t+1}, \pi^t)$$

We assume that every robot i can directly observe the positions and velocities of other robots. Although not all information is observed, the observation vector is determined completely by the given state vector \mathbf{x}^{t+1} .

4.1.6 Transition model $T(\mathbf{x}^{t+1}, \pi^t, \mathbf{x}^t)$

The transition model can be decomposed into a communication step and a physical action step. The communication step is stochastic, only during training (see Section 4.2), and models the effects of communication π^t on the constrained optimization problem used to compute the control actions applied at time step t , \mathbf{u}^t . Then, the robot model f , introduced in Section 3.1, determines the joint state at the next time step \mathbf{x}^{t+1} . Note that, since we are sharing parameters, the communication matrix π^t depends directly on the shared policy. The robots employed in this paper are quadrotors, thus the state transition can be interpreted as the quadrotor model introduced in Appendix A.

4.2 Network Architecture

Given the input (observation \mathbf{z}_i^t) and output (action π_i^t), we elaborate on the communication policy network mapping \mathbf{z}_i^t to π_i^t . We want each robot to process all the information from the environment and decide whether it needs to make a request to any of the other robots. While concatenation of other robots observed information is possible at low scales, learnt policies are bound to quickly deteriorate in performance as the input vector dimensionality grows exponentially with the number of robots. We need an architecture that can provide a compact representation of the observed information of an arbitrary number of other robots, while still being able to leverage that information and choose whether each robot’s trajectory is needed to compute a safe trajectory. Therefore, we need an architecture that can pool together all the information coming from all different robots while still being able to output a different signal for each one of them.

There are several recent works in the field motion planning that use information pooling mechanisms. Often, simulated laser scanner observations allow to consider all information in the

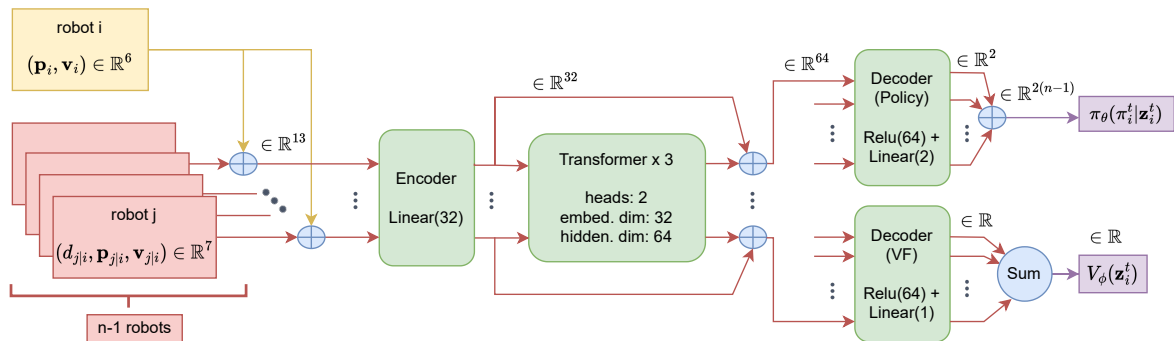


Fig. 2: Proposed network policy architecture. Information from the ego and other robots is marked respectively in orange and red. Red arrows show the flow of information through the architecture of each individual robot j . Architecture layers are marked in green. Concatenation and Sum operations are marked in blue. Outputs are shown in purple.

environment without having to explicitly define each element and its properties (Fan, Long, Liu, & Pan, 2020; R.E. Wang et al., 2020). However, individual information on each of the other agents and their interaction with the environment are lost. Other works address this issue by using Recurrent Neural Networks (RNN) (Hochreiter & Schmidhuber, 1997) over the sequence of other robots observations (Brito, Everett, How, & Alonso-Mora, 2021; Everett et al., 2019). While these methods allow to learn the additional coupled effects resulting from adding each robot into the environment representation, they are not permutation invariant and their performance depends on the heuristic ordering method chosen to feed the elements into the network. Methods using Graph Neural Networks (GNN) (Gama, Marques, Leus, & Ribeiro, 2019), represent elements in the environment as vertices in a graph and allow to learn a permutation equivariant and compact representation of the set of observed information on each one of the vertices (Li, Gama, et al., 2020). In Kurin, Igl, Rocktäschel, Boehmer, and Whiteson (2020), attention mechanisms (Vaswani et al., 2017) are formulated as a GNN for the particular case of fully connected graphs and are used to map a set of sensor measurements from an agent with multiple limbs, to a set of actions to be applied by each one of them. We build our policy network on top of the attention-based architecture proposed in Kurin et al. (2020) and extend its use to homogeneous multi-agent environments.

Our communication policy architecture is depicted in Figure 2. We design a five-hidden-layer neural network as a non-linear function approximation of the policy π_θ . For each robot i , we arrange the information on the other robots $\{\mathbf{d}_{j|i}^t\}_{j \in I \setminus i}$, $\{\mathbf{p}_{j|i}^t\}_{j \in I \setminus i}$, $\{\mathbf{v}_{j|i}^t\}_{j \in I \setminus i}$ into a sequence of vectors $\{(\mathbf{d}_{j|i}^t, \mathbf{p}_{j|i}^t, \mathbf{v}_{j|i}^t)\}_{j \in I \setminus i}$ and append the observed information on robot i at the end of each element in the sequence. Our policy network consists of three parts: an encoder layer, a transformer block and a decoder layer. Each one of these layers have the property of being permutation equivariant and enable processing sequences with an arbitrary number of vectors, even during testing. The encoder layer consists of a linear layer applied independently to each element of the sequence, mapping each element \mathbf{z}_j^t to a latent representation of higher dimension $\tilde{\mathbf{z}}_j^t$. We employ a three-layered transformer (Vaswani et al., 2017), to allow each element $\tilde{\mathbf{z}}_j^t$ in the sequence to exchange information among themselves and encode the information present in the environment while still providing a different result for each element. However, information on each robot’s relative position and velocities are still very important regardless of the additional information and coupled effects coming from other robots in the environment. To preserve this information, we concatenate each element j with its counterpart $\tilde{\mathbf{z}}_j^t$, which also enables the transformer block to focus on learning the coupled effects in communication arising from having multiple agents in the environment.

The network has two decoder heads applied independently to each one of the sequence outputs: one computes the communication action for each robot j : π_i^t , while the other estimates the state-value function: $V^\pi(\mathbf{x}^t) = \mathbb{E}_{x \sim p_{\pi_\theta}, a \sim \pi_\theta} [\sum_{k=0}^{\infty} \gamma^k R_i(\mathbf{x}^k, a) \mid x^0 = x^t]$. Both of them start with a multi-layer perceptron with a hidden layer with a ReLU non-linearity and an output linear layer. The first head outputs a 2-dimensional vector per robot of communication scores that we project onto the probability 1-simplex with a softmax activation function resulting in the vector $[p_{j|i}, 1 - p_{j|i}]$. To enhance exploration, during training the output action is sampled from the resulting Bernoulli distribution $\mathcal{B}(p_{j|i})$. While testing, we follow a deterministic policy where robot i requests j 's trajectory intentions when $p_{j|i} > 0.5$.

The second head outputs a scalar representing the contribution $V_{j|i}$ of each robot j to the value function. Similar to the use of value decomposition networks in collaborative multi-agent tasks (Sunehag et al., 2018), we model agent's i value function as $V^\pi(\mathbf{z}_i^t) = \sum_{j \neq i, j=1}^n V_{j|i}$.

4.3 Multi-Scenario Multi-Stage Training

In order to learn a robust communication policy, we present a multi-stage training scheme in varied scenarios with a clear separation between training and test regime.

4.3.1 Training algorithm

Here we focus on learning a robust communication policy that, in combination with an MPC for motion planning, allows large multi-robot systems to coordinate and navigate at least as safely as when using broadcasting communication policies. To accomplish this, we use the extension to homogeneous multi-agent systems developed in Fan et al. (2020) of the on-policy policy gradient algorithm: Proximal Policy Optimization (PPO) (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) under the assumption of parameter sharing across agents (Gupta et al., 2017), although the general framework is agnostic to the specific RL training algorithm. For this matter, we take the *centralized learning, decentralized execution*

paradigm, which is already popular in multi-agent reinforcement learning for decentralized systems (Everett et al., 2019; Lowe et al., 2017). In particular, the individual policy shared by all agents is learned in a centralized way from the experiences gathered by all robots simultaneously during training. This has been shown to allow the policies of homogeneous agents to be trained more efficiently, and mitigate the non-stationarity in the environment dynamics that arises from having multiple agents learning at the same time. While testing, each robot has copy of the learned policy which is executed in a decentralized fashion.

Algorithm 1 describes the proposed training strategy which alternates between gathering experiences $(\mathbf{z}_i^t, \pi_i^t, R(\mathbf{x}_i^t, \pi_i^t), \mathbf{z}_i^{t+1})$ from all robots and performing PPO gradient updates. PPO is an on-policy method that addresses the high-variance and the difficult hyper-parameter tuning in policy gradient methods for continuous control problems. As suggested in Schulman et al. (2017), in this particular PPO implementation (Liang et al., 2017), we add to the surrogate objective an entropy bonus and a value function loss to ensure sufficient exploration and account for the shared parameters between the policy and the value function. We refer the reader to Schulman et al. (2017) and Fan et al. (2020) for more information on the method's equations and details. The hyperparameters used for training are detailed in table 1.

As explained in Fan et al. (2020), this multi-robot adaptation of the PPO algorithm can be parallelized and easily scaled to large-scale multi-robot systems since every robot counts as an independent worker gathering data. This reduces the sampling time cost and makes the algorithm suitable for training a large number of robots in various scenarios, profiting from frameworks specialized in distributed computation (i.e. Ray (Moritz et al., 2018) and RLlib (Liang et al., 2017)) to accelerate our network's convergence.

4.3.2 Training scenarios

While proper exploration of the action and state spaces is crucial for the quality, robustness and generalization characteristics of the learned communication policy, it is difficult to achieve proper exploration of the state space since our policy only decides on *whom to communicate with*. Therefore,

Algorithm 1 PPO for multiple agents with parameter sharing

```

1: Initialize policy network  $\pi_\theta$  and value function  $V_\phi$ . Set hyper-parameters as shown in Table 1. Note
   that  $\theta$  and  $\phi$  share the same set of parameters except for the decoder layer.
2: for iteration = 1,2,..., do
3:   for Robot  $i = 1,2,\dots,n$  do
4:     // Collect data in parallel. We define  $r_i^t = R_i(\mathbf{x}_i^t, \pi_i^t)$ 
5:     for  $e = 1,2,\dots,n_e$  do
6:       Run comm. policy  $\pi_\theta$  for episode  $e$ , collecting  $\{\mathbf{z}_i^t, r_i^t, \pi_i^t\}$  where  $t \leq T_e$  (Algorithm 2)
7:       Estimate and collect advantages using GAE (Schulman, Moritz, Levine, Jordan, & Abbeel,
2016):  $\hat{A}_i^t = \sum_{l=0}^{T_e-t} (\gamma\lambda)^l \delta_i^{t+l}$  where  $\delta_i^t = r_i^t + \gamma V_\phi(\mathbf{z}_i^{t+1}) - V_\phi(\mathbf{z}_i^t)$ 
8:       Estimate and collect target values:  $V_{target}^t(\mathbf{x}_i^t) = \sum_{t'>t}^{T_e} \gamma^{t'-t} r_i^{t'}$ 
9:     end for
10:   end for
11:    $\pi_{old} \leftarrow \pi_\theta$ 
12:   //Update policy and value function
13:   for  $j = 1,\dots,E_\pi$  do
14:     for  $b = 1,\dots,(n_e T_e n) // n_{minibatch}$  do
15:       Sample minibatch  $D_b$  from collected rollout data  $D$ 
16:       //Surrogate objective. We define:  $h_i^i(\theta) = \frac{\pi_\theta(\pi_i^t | \mathbf{z}_i^t)}{\pi_{old}(\pi_i^t | \mathbf{z}_i^t)}$ 
17:        $L^O(\theta) = \mathbb{E}_{(\mathbf{z}_i^t, r_i^t, \pi_i^t, V_{target}^t(\mathbf{x}_i^t), \hat{A}_i^t) \sim D_b} [\min(h_i^i(\theta) \hat{A}_i^t, \text{clip}(h_i^i(\theta), 1-\epsilon, 1+\epsilon) \hat{A}_i^t) + \beta KL[\pi_{old} | \pi_\theta])]$ 
18:       //Value function Loss
19:        $L^{VF}(\phi) = \mathbb{E}_{(\mathbf{z}_i^t, r_i^t, \pi_i^t, V_{target}^t(\mathbf{x}_i^t), \hat{A}_i^t) \sim D_b} [(V_{target}^t(\mathbf{x}_i^t) - V_\phi(\mathbf{z}_i^t))^2]$ 
20:       //Entropy objective. We define  $S[\pi_\theta](\mathbf{z}_i^t)$  as the entropy of the policy distribution  $\pi_\theta(\pi_i^t | \mathbf{z}_i^t)$ 
21:        $L^S(\theta) = \mathbb{E}_{(\mathbf{z}_i^t, r_i^t, \pi_i^t, V_{target}^t(\mathbf{x}_i^t), \hat{A}_i^t) \sim D_b} [S[\pi_\theta](\mathbf{z}_i^t)]$ 
22:       //Total PPO objective
23:        $L^{PPO}(\theta, \phi) = L^O(\theta) - c_1 L^{VF}(\phi) + c_2 L^S(\theta)$ 
24:       Update policy param.  $\theta$  with  $lr_\theta$  by Adam (Kingma & Ba, 2015) with respect to  $L^{PPO}(\theta, \phi)$ 
25:     end for
26:   end for
27:   //Adapt KL penalty coefficient
28:   if  $KL[\pi_{old} | \pi_\theta] > 2KL_{target}$  then
29:      $\beta \leftarrow 1.5\beta$ 
30:   else if  $KL[\pi_{old} | \pi_\theta] < 0.5KL_{target}$  then
31:      $\beta \leftarrow \beta/2$ 
32:   end if
33: end for

```

it is necessary to design interaction-rich training scenarios where the robots can sample meaningful experiences that will allow them to learn when it is necessary to cooperate and request other robots future trajectory intentions.

We have created a simulation environment (**Zhu & Alonso-Mora, 2019b**) where a group of twelve drones navigate from an initial position to a goal position and may communicate their trajectory plans to perform collision avoidance. We have designed three different scenarios to train our

communication policy, as depicted in the left column of **Figure 3**. Each scenario requires increasing levels of interaction and cooperation to perform collision avoidance, ranging from a simple scenario where almost no communication is needed (e.g., **Figure 3a**) to complex scenarios where the drones must communicate (e.g., **Figure 3c**) to successfully avoid each other. The employed scenarios are:

- **Random navigation (Figure 3a)**: Each robot must to move to a random goal position.

Table 1: Hyperparameters for PPO training algorithm

Parameter	Value
Lambda λ	1.0
Gamma γ	0.99
Episodes each iteration n_e	40
Episode time steps/episode T_e	100
Number of epoch per iteration E_θ	30
SGD minibatch size $n_{minibatch}$	512
Clip param. ϵ	0.3
KL target KL_{target}	0.01
Learning rate lr_θ	5e-5
KL coeff. β	0.2
Value function loss coeff. c_1	1
Entropy loss coeff. c_2	0.001
Gradient Clipping	0.1

- **Random swapping (Figure 3c):** The group of robots is arranged in pairs. Then each robot switches position with its counterpart.
- **Asymmetric swapping (Figure 3e):** We split the \mathbb{R}^2 x-y plane into twelve quadrants and randomly initialize each robot in a different quadrant with random initial position. Then, each robot swaps positions with a robot from the diametrically opposed quadrant.

The learned communication policy depends on the scenarios on which it is trained. For instance, if an agent is trained only on the first scenario it will learn a *no communication* behavior. In contrast, if only trained in the last one, it may learn to always communicate. Hence, we employ *curriculum learning* (Bengio, Louradour, Collobert, & Weston, 2009), training the agents first in a simple scenario, where communication is generally not needed, and subsequently introducing more difficult and complex scenarios where the agents must learn *when* and *with whom* to communicate. We design the learning process in three stages of 12500 episodes, through which we sample episodes from a scenario pool. During the first stage the scenario pool only has the first type of scenario, thus we sample random navigation scenarios with probability one. In the second step, we add the second type to the scenario pool, sampling the new scenario 75% of times. Finally, during the last stage we add the *asymmetric swapping* type to the pool, sampling this one 75% of times and 12.5% each of the others.

4.3.3 Training/Test regime differentiation

As explained in section 4.3.2, the learned communication policy cannot explore directly the state space since only the MPC is in charge of motion planning. Also, communication is not the only source of cooperation as the other robot’s trajectory intentions can be approximated by constant velocity models. This results in our robots rarely exploring collisions or dangerous situations even when not communicating, thus creating sparsity and a lack of collision experiences to learn from. On top of this, while all communications are punished instantaneously when they happen, there is a delay between issuing a communication request (or not) and its associated reward for completing the episode successfully (or colliding).

We apply a crucial change between the training and testing regimes to promote exploration of the state space, specially collision events. During training, we turn off constant velocity predictions whenever robot i does not communicate with robot j to request its trajectory intentions. This makes robot i MPC planner virtually blind to robot j future intended positions. This will ultimately result in a learnt communication policy that keeps track and decides to request trajectory intentions from all robots that might put the safety of our future trajectory at risk, or need to be cooperated with. As seen in section 5.6.3, this modification results in an efficient and intuitive communication policy that is as safe as broadcasting policies. During testing, we turn on once again trajectory estimations whenever there is no communication both as a safety layer and to avoid generating oscillating trajectories due to switching on and off constraints related to other robots’ future trajectories in the MPC.

The main idea of turning robot i blind to robot j whenever it does not request j ’s trajectory intentions, is to force collisions to happen whenever necessary communications are not effectuated. This way, it is easier to discriminate and learn the cause and effect relationship between a communication signal being (not) triggered between i and j and their subsequent collision (not) being avoided. The intuition behind this training approach is to learn *with whom to cooperate* rather than *with whom to communicate* as it results in a policy that puts the attention in those other robots in the

team that need to be cooperated with to achieve safe navigation. Although less information for the motion planner during training results in a suboptimal policy for testing, i.e. more communicative than necessary, we argue that the additional information can help in some situations and, at worst, will not make the resulting policy less safe. This is why the found solution is able to achieve similar performance to broadcasting policies in terms of safety. The robot motion planning loop during training and testing are detailed in Algorithm 2.

4.4 Predicting and Generating Safe Trajectory Intentions

4.4.1 Informed constant velocity estimations

At time step t , robot i can obtain $\hat{\mathcal{T}}_{j|i}^t$ either by communicating and requesting robot j 's future trajectory intention ($\hat{\mathcal{T}}_{j|i}^t = \mathcal{T}_j^{t-1}$) or directly by computing an estimation of said trajectory intentions ($\hat{\mathcal{T}}_{j|i}^t = \text{prediction}(\mathbf{x}_j^t)$). Algorithm 3 depicts the structure of the prediction function.

Whenever robot i requests robot j 's trajectory intentions at time step t , it stores both the last communicated trajectory \mathcal{T}_j^{t-1} and the time step of communication t . Every subsequent time step $t+k$, we make sure the last communicated trajectory is not obsolete, and that robot j is within a predefined tolerance region around its trajectory intention r_{tol} . If any of these conditions is not true, the last communicated trajectory is discarded and robot j 's future positions are estimated by assuming it will follow constant velocity. If the communicated information is not discarded, we take the remaining $N-1-k$ steps from the tail of the communicated trajectory, and expand them until obtaining a set of N future positions by assuming constant velocity.

4.4.2 Generating safe trajectory intentions

At time t , given robot i 's requests for information π_i^t , we can determine $\hat{\mathcal{T}}_{j|i}^{t-1}$. Then, robot i 's computed trajectory intentions \mathcal{T}_i^t and control inputs \mathbf{u}_i^t are computed by solving a constrained optimization problem. This optimization problem computes the optimal future values for $\{(\mathbf{x}_i^{t+l+1}, \mathbf{u}_i^{t+l}) \mid \forall l = 0, \dots, N-1\}$, that minimize, over a N -time step horizon, a given cost

Algorithm 2 WW2C framework

- 1: **Inputs:** Number of robots n . Starting and goal positions and initial velocities: $\{\mathbf{p}_i^0, \mathbf{p}_{i,g}, \mathbf{v}_i^0\}, \forall i \in \mathcal{I}$. Training/Testing communication policy. n number of robots. Episode length: T_e . Maximum velocity: \mathbf{v}_{max} .
- 2: **Initialize** n robots in their initial positions and velocities. Copy an instance of the learned/still in training communication policy π_θ to all robots.
- 3: **Initialize** within each robot a set of last communicated trajectories $\{D_j\}_{j \neq i}$. Obtain the first observation of the environment $\{\mathbf{z}_i^0\}_{i \in \mathcal{I}}$
- 4: **for** time step $t = 0, 1, \dots, T_e$ **do**
- 5: **for** robot $i = 1, 2, \dots, n$ **do**
- 6: //Compute $\pi_i^t = \{c_{j|i}^t\}_{j \neq i}$
- 7: **if** training **then**
- 8: $\pi_i^t \sim \mathcal{B}(\pi_\theta(\mathbf{z}_i^t))$
- 9: **else**
- 10: $\pi_i^t = \mathbb{1}[\pi_\theta(\mathbf{z}_i^t) > 0.5]$
- 11: **end if**
- 12: //Compute $\hat{\mathcal{T}}_{j|i}^{t-1}$
- 13: **for** robot $j \neq i, i = 1, \dots, n$ **do**
- 14: **if** $c_{j|i}^t == 1$ **then**
- 15: //Traj. intention requested
- 16: $\hat{\mathcal{T}}_{j|i}^{t-1} = \mathcal{T}_j^{t-1}$
- 17: $D_j \leftarrow \mathcal{T}_j^{t-1}$
- 18: **else if not** training **then**
- 19: //Traj. int. predicted (alg. 3)
- 20: $\mathcal{T}_j^{t-k-1} \leftarrow D_j, k > 0$
- 21: $\hat{\mathcal{T}}_{j|i}^{t-1} = \text{pred}(\mathcal{T}_j^{t-k-1}, \mathbf{v}_j^t, \mathbf{p}_j^t)$
- 22: **else**
- 23: $\hat{\mathcal{T}}_{j|i}^{t-1} = \emptyset$
- 24: **end if**
- 25: **end for**
- 26: $\{\mathcal{T}_i^t, \mathbf{u}_i^t\} \leftarrow \text{Solve eq.11 with } \mathbf{g}_i = \mathbf{p}_{i,g}$
- 27: **end for**
- 28: $\{\mathbf{z}^{t+1}\}_{i \in \mathcal{I}} \leftarrow \text{Step}(\{\mathbf{u}_i^t\}_{i \in \mathcal{I}})$
- 29: **end for**

function (defined in Appendix B). The solution of the problem is constrained to follow the robot's dynamic model \mathbf{f} and account for the estimates of other robots' trajectory intentions $\hat{\mathcal{T}}_{j|i}^{t-1}$ to avoid future collisions (equality and inequality constraints). The sequence of intended future positions in $\{\mathbf{x}_i^{t+l+1} \mid \forall l = 0, \dots, N-1\}$ is used to construct \mathcal{T}_i^t , while only the first value of the sequence $\{\mathbf{u}_i^{t+l} \mid \forall l = 0, \dots, N-1\}$, \mathbf{u}_i^t , is used.

Table 2: Hyperparameters for the WW2C framework

Parameter	Value
Episode length T_e	100
Trajectory prediction horizon N	20
Time step length Δt	0.05s
Tolerance length r_{tol}	0.1m

The sequence $\{(\mathbf{x}_i^{t+l+1}, \mathbf{u}_i^{t+l}) \mid \forall l = 0, \dots, N - 1\}$ is computed at every time step t by formulating and solving the following constrained optimization problem:

$$\begin{aligned}
& \min_{\mathbf{x}_i^{0:N}, \mathbf{u}_i^{0:N-1}} \sum_{k=0}^{N-1} J_i^k(\mathbf{x}_i^k, \mathbf{u}_i^k) + J_i^N(\mathbf{x}_i^N, \mathbf{g}_i) \\
& \text{s.t. } \mathbf{x}_i^0 = \mathbf{x}_i^t, \\
& \quad \mathbf{x}_i^{k+1} = \mathbf{f}(\mathbf{x}_i^k, \mathbf{u}_i^k), \\
& \quad \left\| \mathbf{p}_i^{k+1} - \hat{\mathbf{p}}_{j|i}^{k+1} \right\| \geq 2r, \quad \forall j \in \mathcal{I} \setminus \{i\} \\
& \quad \mathbf{u}_i^k \in \mathcal{U}, \quad \mathbf{x}_i^{k+1} \in \mathcal{X},
\end{aligned} \tag{11}$$

where $\hat{\mathbf{p}}_{j|i}^{k+1}$ are extracted from $\hat{\mathcal{T}}_{j|i}^{t-1}$. $J_i^k(\mathbf{x}_i^k, \mathbf{u}_i^k)$ and $J_i^N(\mathbf{x}_i^N, \mathbf{g}_i)$ are the stage and terminal cost functions to be minimized, which are defined in Appendix B. Function \mathbf{f} is the non-linear discrete function representing the dynamic model of the robot.

5 Simulation Experiments

In this section we first describe our implementation of the proposed method. Next, we provide a thorough evaluation of our learned communication policy by comparing it with our previous approach (Serra-Gómez et al., 2020) and other communication baselines in several scenarios requiring increasing cooperation efforts to navigate safely. A video demonstrating the results of this paper is available (see Extension 1).

5.1 Training Setup

We train our communication policy for a team of twelve quadrotors moving in \mathbb{R}^3 and following Parrot Bebop 2 dynamics. We rely on the solver Forces Pro (Domahidi & Jerez, 2014) to generate optimized NMPC code and its corresponding Python wrapper. As in Zhu, Juhl, Ferranti, and Alonso-Mora (2019), the time step used in the

NMPC is 0.05s and the prediction horizon is $N = 20$ (1 second ahead). The constraints are formulated as soft-constraints to ensure the feasibility of the problem, and the solver iterations have been limited to 600 to have at least a control frequency of 20 Hz. Note that the framework is agnostic to the choice of solver as long as it allows a control frequency of 20 Hz. The learning algorithm and the training of our policy were implemented in Tensorflow, using the RLlib framework (Liang et al., 2017). The Critic and Actor models follow the architecture shown in section 4.2 and were trained for 37200 episodes using an Intel i9-9900 CPU@3.10GHz computer. The hyperparameters used for training are explained in the Table 1. The simulation time step is set to 0.05s, which is the robot’s control period. The quadrotors’ dimensions are represented by a sphere of radius $r = 0.3m$ and their maximum speed is $v_{max} = 4.25m/s$. Computing both the communication policy and the MPC control inputs takes less than 0.01s per robot for each time step, which allows for a real-time implementation of the framework with a control and communication frequency of 20Hz. No noise is added into the simulation environment during the training process, in order to optimize the policy with low variance. Values for the reward weights were $w_g = 10$, $w_{coll} = 10$, $w_c = 10$. Tuned reward and penalty terms were $r_g = 1$, $r_{coll} = 1$ and $N_i(n) = 100(n - 1)$. Goal reward is only received once during the episode. Episodes are finished after reaching 100 time steps or when all agents reach the goal.

5.2 Baselines

We introduce and compare our method with four other commonly used heuristic communication policies:

- *Full communication* (FC): At each time step each robot broadcasts its trajectory plans.
- *No communication* (NC): The robots never exchange their trajectory plans and a Constant Velocity model is used by each robot to infer the others trajectories.
- *A distance-based communication policy* (ϵ -DBCP): If the distance between two robots distance is smaller than a threshold ϵ (in meters) then the agents broadcast their trajectory information. $\epsilon \in \{4.25m, 8.5m\}$, which is once and

Algorithm 3 Informed constant velocity estimator of robot j

```

1: Inputs: Last communicated trajectory at time step  $t - k$ :  $\hat{\mathcal{T}}_{j|i}^{t-k-1}$ ,  $j$ 's current velocity and position:
    $\mathbf{p}_j^t$   $\mathbf{v}_j^t$ . Hyperparameters in table 2.
2: reject = 0
3: if  $k \geq N - 1$  then
4:   reject = 1
5: else if  $\|\hat{\mathcal{T}}_{j|i}^{t-k-1}(k) - \mathbf{p}_j^t\| > r_{tol}$  then
6:   reject = 1
7: end if
8: if reject then
9:    $\hat{\mathcal{T}}_{j|i}^{t-1} = \{\hat{\mathbf{p}}_j^{t+l} \mid \hat{\mathbf{p}}_j^{t+l+1} = \hat{\mathbf{p}}_j^{t+l} + \Delta t \mathbf{v}_j^t, l = 0, \dots, N - 1; \hat{\mathbf{p}}_j^t = \mathbf{p}_j^t\}$ 
10: else
11:   //Expand the tail of the last communicated trajectory. Note that we consider:  $\hat{\mathcal{T}}_{j|i}^{t-k-1}(0) = \mathbf{p}_j^{t-k-1}$ 
12:    $Tail = \{\hat{\mathbf{p}}_j^{t+l} \mid \hat{\mathbf{p}}_j^{t+l} = \hat{\mathcal{T}}_{j|i}^{t-k-1}(k + 1 + l), l = 0, \dots, N - k - 2\}$ 
13:    $step_{cte.v} = \hat{\mathcal{T}}_{j|i}^{t-k-1}(N - 1) - \hat{\mathcal{T}}_{j|i}^{t-k-1}(N - 2)$  //Cte. velocity step at the end of the tail
14:    $\hat{\mathcal{T}}_{j|i}^{t-1} = Tail \cup \{\hat{\mathbf{p}}_j^{t+l} \mid \hat{\mathbf{p}}_j^{t+l+1} = \hat{\mathbf{p}}_j^l + step_{cte.v}; l = N - k - 1, \dots, N - 1; \hat{\mathbf{p}}_j^{t+N-k-2} = \hat{\mathcal{T}}_{j|i}^{t-k-1}(N - 1)\}$ 
15: end if
16: return  $\hat{\mathcal{T}}_{j|i}^{t-1}$ 

```

twice the maximum distance within planning horizon, respectively.

Full communication and *no communication* policies give us a reference on what are the expected maximum and minimum performances in terms of safety and communication requests. On the one hand, since *full communication* policies allow each robot to request trajectory intentions from all robots at every time step, we can consider it to be an over-conservative communication policy. Thus, if safe navigation is not achievable by applying *full communication* in a particular sampled episode, we can consider that it is difficult to find a better communication policy that can achieve collision avoidance for this particular configuration of robot initial positions and goals. On the other hand, *no communication* policies provide a reference on the expected minimum performance of the framework when only constant velocity estimations are used to predict other robots trajectory intentions. Other baselines (ϵ -DBCP) give us a sense of our learned method's efficiency and safety in comparison with hand-crafted, reasonable and strong heuristics. The MPC motion planner is implemented with the same parameters for all baselines.

5.3 Testing Scenarios

To evaluate and compare our method with the baselines we design scenarios where we can evaluate how communication policies adapt to different levels of interaction. Therefore, aside from the scenarios used for training (see Sec. 4.3.2), we define three additional ones:

- **Rotation (Figure 3b):** All drones are arranged in a circle and must rotate one position either clockwise or counter-clockwise. This is a control scenario where no communication should be necessary. Therefore it allows us to evaluate the adaptability and communication efficiency of the policy.
- **Group swapping (Figure 3d):** We arrange the twelve drones in two groups of six symmetrically opposed. Then, each drone must swap positions with its symmetrical counterpart.
- **Symmetric swapping (Figure 3f):** All drones are arranged in a circle in symmetrically opposed initial positions and swap places with the opposite drone. As with the asymmetric swapping scenario, communication is required for all drones to ensure collision-free trajectories.

All scenarios used for evaluation are depicted in Figure 3.

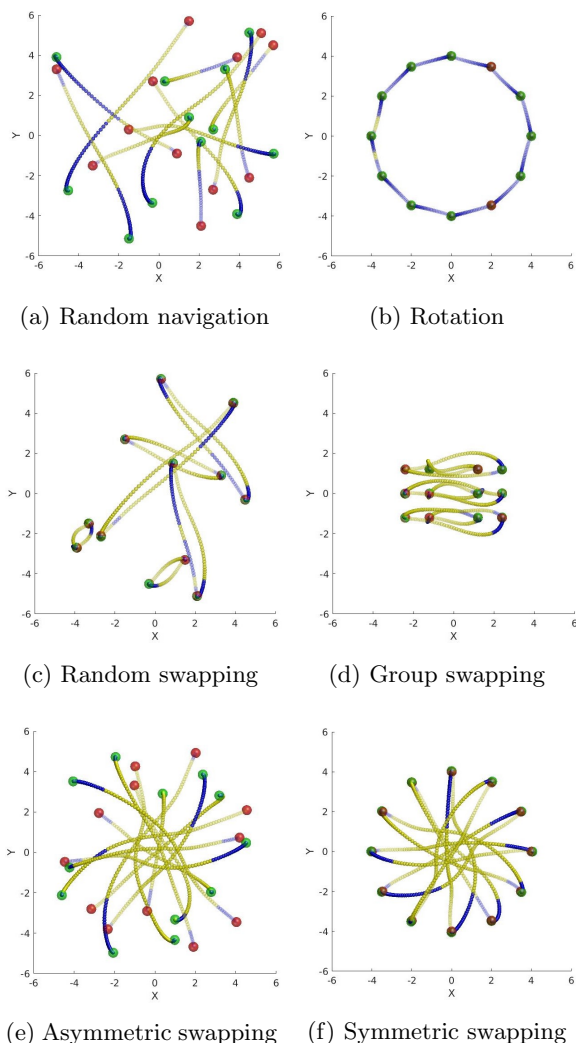
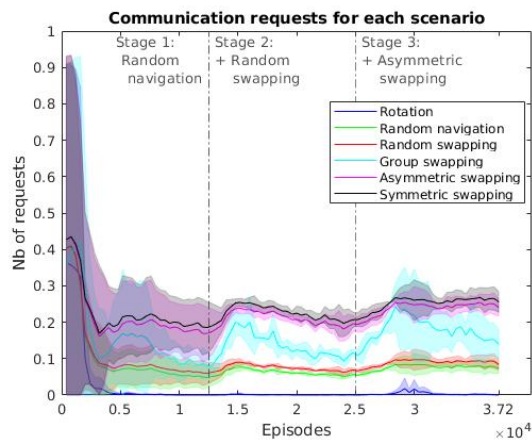


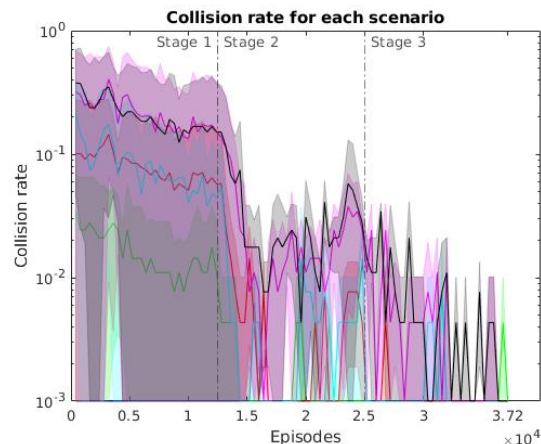
Fig. 3: Simulation results for each scenario using our communication policy. The three figures on the left show the scenarios used for training while the three on the right are the ones used for testing. Solid lines represent the trajectories executed by the drone-swarm. Yellow represents the positions where the drones communicate their trajectory plans. Blue depicts the positions where the drones do not communicate. Green and Red represent the initial and goal position of each drone, respectively. Increasing opacity represents the episode progression.

5.4 Performance Evaluation

We evaluate the performance of the proposed learned collision-avoidance policy in terms of its



(a) Evolution throughout training of the learned policy's number of communication requests.



(b) Evolution throughout training of the learned policy's collision rate. Since the y-axis is in logarithmic scale, the 0 value is represented by 10^{-3} .

Fig. 4: Policy evolution throughout the training. We train three seeds and evaluate them every 400 training episodes for 100 episodes. We show the evolution of the mean and standard deviation of their performance in each of the presented scenarios. Communication requests are normalized by the results that would be obtained for the *full communication* policy. We use the results obtained for the full communication policy to normalize the number of communication requests of the learned policy between $[0,1]$.

safety and communication efficiency. We present multiple performance metrics and then compare

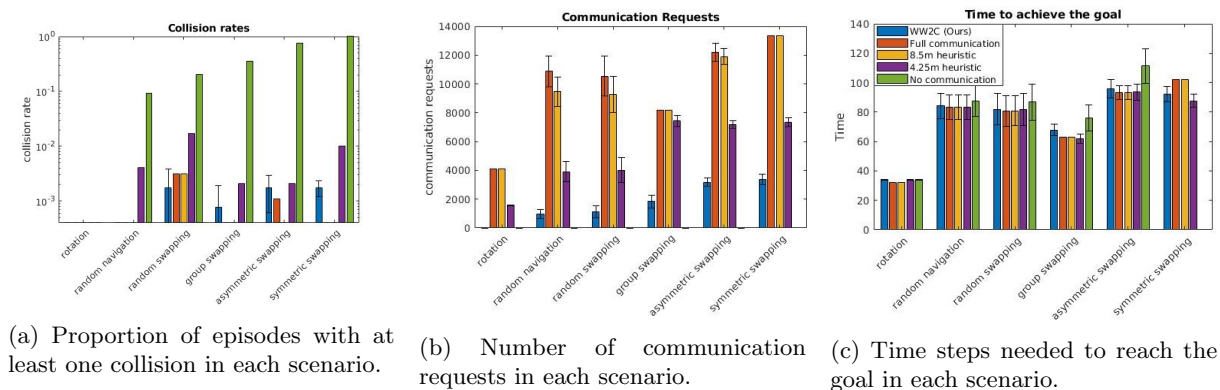


Fig. 5: Performance evaluation for each scenario of the communication baselines and our learned policy. For our trained policy, we run three seeds and take their average performance and standard deviation. We run each method for 1000 episodes, gathering results for each metric. For collision rates, we show the proportion of episodes where we obtain at least one collision. For communication requests and number of time steps, we show the results for those episodes without collisions. In case none of the sampled episodes end without collisions, no bar is shown for that policy.

our method with the indicated baselines. The metrics are:

- *Collision rate:* Proportion of episodes where there has been a collision between any of the robots in the team.
- *Number of communication requests along the episode:* Total number of communication requests throughout the episode. In deterministic scenarios, where we are certain of the low or high need for cooperation, this metric will allow to discern the adaptability of each model.
- *Time to achieve the goal:* Number of time steps needed to reach the goal. Failed episodes, where a collision has happened, are not accounted for when computing the mean and standard deviation.

Fig 4 shows the evolution of the learned policy throughout training in terms of average collision rate and number of communication requests for all scenarios. To account for the effect of different network initialization seeds into the final learned policy, we train three different initialization seeds and show the average and standard deviation of their performance in all our evaluations.

We normalize our results in communication requests between $[0,1]$ using those obtained for the *full/no communication* policies for the same sets of sampled episodes for each scenario since this gives us a reference on the maximum/minimum

values that we can score in both metrics. We can see that our method is able to learn an adaptable policy that makes close to no requests in simple settings such as *Rotation*, *Random navigation* and *Random swapping* scenarios while marginally affecting the number of collisions obtained in more difficult settings such as *Group swapping*, *Asymmetric* and *Symmetric swapping* scenarios. We can observe that our learned policy can adapt to the different amounts of communication that are required to achieve safe navigation. Note how at the end of training, the collision rate decreases drastically.

Figure 5 compares the learned communication policy against the proposed baselines using these metrics. The scenarios are ordered according to their levels of interaction. The results obtained for the *no communication* policy show correlation between the different complexity in scenarios and the need for communication. As shown in Figure 5a, the collision percentage of our method (WW2C) is the same or very similar as all other conservative baselines in all scenarios (0.4% difference at most). Even in the most complex scenarios, such as *asymmetric swapping*, we show that the difference in collisions is not significant in comparison with the safest communication policy: *full communication*. The clear advantage of our method is illustrated in Figure 5b, where WW2C shows better results by communicating less than

50% and 30% in comparison with *4.25m heuristic* and the *full communication* in the worst cases. We show that the learned policy can adapt better to scenarios of different complexities since there is also a clear correlation between the amount of communication requested at each scenario and the expected need for cooperation shown in Figure 5a.

Due to our setup not allowing our drone to stop instantaneously, it will collide rather than run into deadlocks. Therefore, the success rate for any method is equivalent to one minus its collision rate. We show that the learned communication policy still manages to succeed in practically all scenarios with little effect in the time it needs to achieve the goal in comparison with the baselines. Our reward function accounts for achieving the goal only at the end of the episode. This is due to our main priority being to decrease the amount of communication while maintaining safety and avoiding deadlocks. We find that, although our reward function does not motivate achieving the goal as fast as possible, the sacrifice in terms of additional time steps is not significant.

5.5 Robustness and Zero-Shot Generalization Capabilities

Our approach allows to obtain a policy that is capable to generalize to an arbitrary number of robots in the environment, and to scenarios requiring different levels of interaction. Thus, we demonstrate the generalization of the learned communication policy with a series of experiments.

5.5.1 Lower/Larger scale multi-robot systems

We evaluate the performance of our method trained with 12 agents, on scenarios with a higher/lower number of agents. More specifically, in Figure 6, we show the obtained results from simulating 6,12,18 and 24 robots in each of the training and testing scenarios for 1000 episodes in comparison to the performance shown by the *full communication* policy under the same conditions.

We show that our method is able to communicate at least 70% less in comparison with the *full communication* policy, while still being capable to adapt to scenarios requiring different levels

of interaction. Note that the normalized communication requests for each scenario does not change significantly with the number of agents, even showing a decreasing tendency when scaling up to 24 agents. This indicates that our communication policy generalizes well to environments with additional agents.

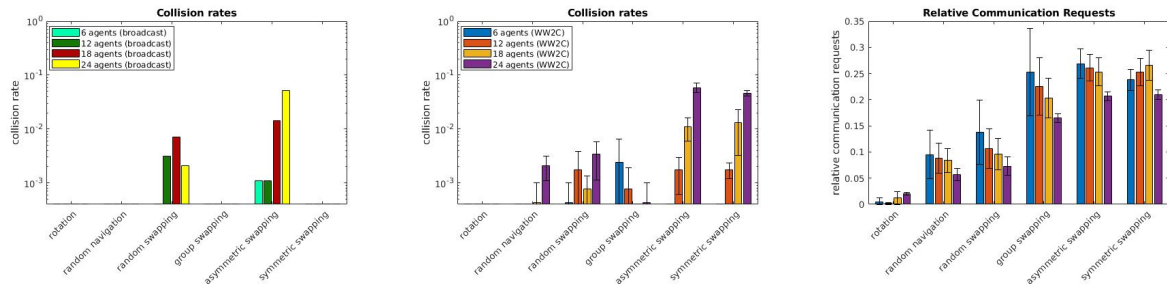
Regarding the obtained collision rates, our method generalizes well and shows better results when there are less robots in the environment. There is a degradation of performance when the number of agents in the environment is higher than seen during training. However, the degradation obtained for our method is low (i.e., less than 2% for 18 agents, and less than 10% for 24 agents) and is similar to the degradation seen when using *full communication* for the same numbers of agents.

5.5.2 Noisy positions and velocities

We also evaluate the robustness of our communication method under different levels of noisy inputs. We add a multiple of a gaussian noise to the other agents' relative positions and velocities in our observation vector \mathbf{z}_i^t to simulate the effects of sensor measurement errors and localization uncertainties on our learned communication policy. The added measurement noise is zero mean with covariance: $\Sigma = \text{diag}(0.06\text{m}, 0.06\text{m}, 0.06\text{m})^2$. We simulate 1000 episodes for each scenario under three levels of noise: Σ , 2Σ , 4Σ . In figure 7, we show that our method is robust to these different levels of the added measurement noise since both performance and behaviour in terms of collision rates and communication requests suffer non-significant changes. In fact, note how the collision rates for each level of noise remains very low ($> 1\%$) and similar to the other results.

5.6 Ablation Study

We analyse the key design choices we have introduced in this paper in comparison to [Serra-Gómez et al. \(2020\)](#). Two main changes that we introduce are a model architecture that is able to function with an arbitrary number of robots, and a difference in conditions between training and testing regimes to obtain more robust and adaptable communication policies. Overall, these two changes allow to learn policies that can decide better *when* and *with whom* to communicate. We also changed

(a) Collision rate over each scenario for *full communication* policy.

(b) Collision rate over each scenario for our method.

(c) Normalized number of communication requests over each scenario.

Fig. 6: Results obtained when testing the *full communication* and our policy with 6, 12, 18 and 24 drones. Our method has been trained with 12 drones. Shown communication requests are have been scaled using the mean communication requests of the *full communication* policy under each scenario/number of agents.

the reinforcement learning algorithm from MADDPG (Lowe et al., 2017) to PPO (Schulman et al., 2017) with parameter sharing (Gupta et al., 2017). This was necessary as MADDPG requires one state-action value function per agent, which scales badly with the number of agents and tends to learn specialized agent roles that are situation specific. PPO, on the other hand, only requires one state-value function for all agents and can learn the same communication policy for all agents using parameter sharing as well, avoiding over-specialization to specific scenarios.

We perform an ablation study to justify the modifications applied to the previous approach (Serra-Gómez et al., 2020). First, we will address the implementation of a targeted scalable attention-based architecture to encode the information of the dynamic environment. Second, we will empirically justify our decision of disabling informed constant velocity estimations whenever there’s no communication during training.

5.6.1 Evaluation metrics and output models

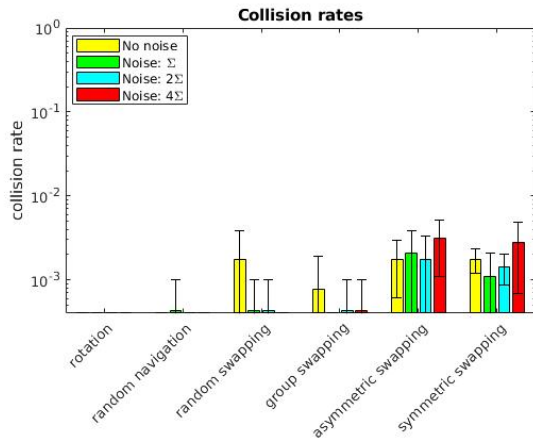
We compare the different ablations across the same scenarios mentioned in section 5.4 showing the evolution of their training according to the following metrics (sorted in descending priority order):

- *Collision rate*: Defined in section 5.4.
- *Number of communication requests*: Defined in section 5.4.

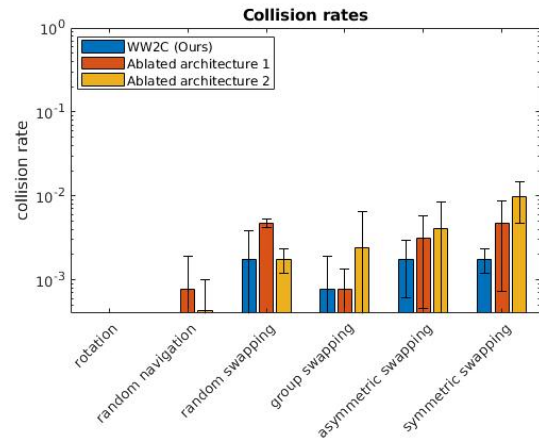
Similar to our method in section 4.2, each ablated communication policy outputs a normalized 2-dimensional vector $[p_{j|i}, 1 - p_{j|i}]$ for each other robot j . This vector represents a Bernoulli distribution $\mathcal{B}(p_{j|i})$. The communication policy is stochastic if it samples such distribution to decide whether to request robot j ’s trajectory intentions ($c_{j|i} \sim \mathcal{B}(p_{j|i})$). The policy is deterministic if it decides to communicate by comparing the mean of the distribution to a predetermined threshold ($c_{j|i}^t = \mathbb{1}[p_{j|i} > 0.5]$). For fair comparison, we evaluate every ablation both as a stochastic and deterministic policy and show the one obtaining the best performance across all different evaluation scenarios.

5.6.2 Scalable attention-based architecture

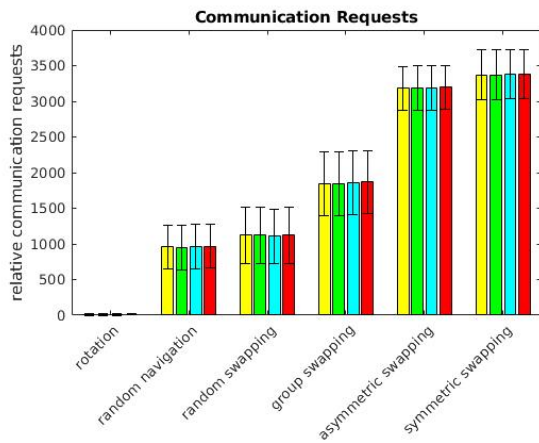
One of the main limitations of our previous approach (Serra-Gómez et al., 2020) is its difficulty to scale to multi-robot teams larger than four quadrotors, let alone react to an arbitrary number of robots in the environment. Similar to Everett et al. (2019) and Kurin et al. (2020), in this work we address this challenge by incorporating three layers of transformer blocks into the core of the network architecture to encode the environment, which allows to provide a communication action for an arbitrary number of other robots. In Figure 8, we compare our approach with two ablated versions.



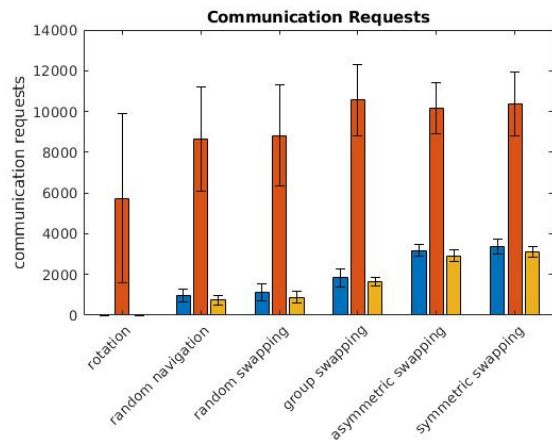
(a) Collision rate over each scenario.



(a) Collision rate



(b) Number of communication requests over each scenario.



(b) Number of communication requests

Fig. 7: Results obtained when testing our policy in presence of noise of the observed robot’s positions and velocities. Standard deviations are taken over the results taken when evaluating 3 seeds of the trained policy over 1000 episodes.

Ablated architecture 1 concatenates all the encoded vectors from other robots and substitutes the transformer block by three 64 neuron fully-connected layers with a ReLU activation. The decoder layer maps the resulting hidden layer to a vector of $2(n - 1)$ communication scores. This ablated version cannot be used for a different number of agents than in training. The aim of this ablated version is to showcase the benefits of using each other drone individual information while using attention mechanisms to encode the

Fig. 8: Performance evaluation over the ablated versions presenting different policy architectures. We add the performance of our own method for comparison. Standard deviations are taken over the results taken when evaluating 3 seeds of each trained policy over 1000 episodes.

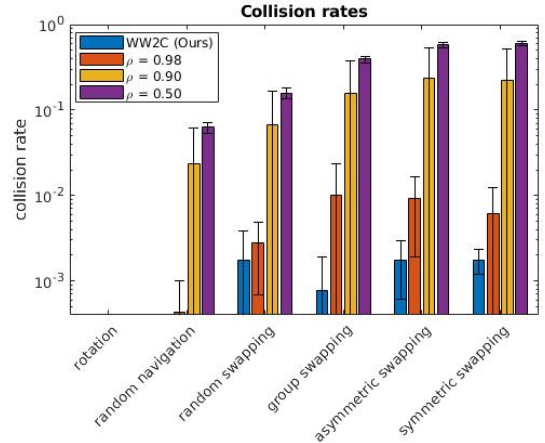
state of the environment and compute each communication signal. In Figure 8, we show that the architecture used in this work is able to scale better to larger multi-robot systems both in terms of collision rate and number of communication requests across episodes. These results also remark the importance of precise communication. A higher amount of communication requests do not necessarily translate to a safer communication policy.

Ablated architecture 2 also replaces the transformer block with three fully-connected layers of 64 neurons with ReLU activation functions. However, it processes each robot’s information individually to decide whether to communicate or not. This network solves a simpler problem than the precedent ablated version since it learns to communicate with another other robot by only considering the information on its distance, relative position and velocity. Surprisingly, Figure 8 shows that this second ablation performs similar to our attention-based architecture. The intuition behind these results is that, at least in the tested scenarios, the individual relative information of every other robot contains most of the information that is relevant to decide whether to communicate with it or not. Although our method seems to result in slightly less collisions in all scenarios, we cannot draw a solid conclusion on this since the performance is not significantly different. In fact, while it is logical that there should be situations where the attention module would add a clear advantage over the pairwise communication ablation, it seems difficult to identify and reproduce these scenarios.

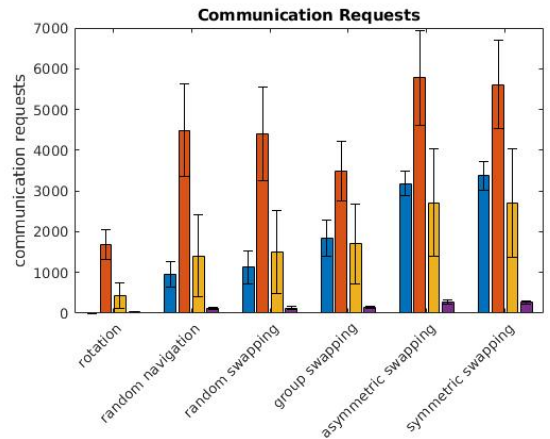
5.6.3 Training-test environment separation

As explained in section 4.3.3, distinction between test and training regimes was applied to increase the amount of collision experiences and increase the causality between lack of communication and collision events. The result of doing this is an efficient learned communication policy that is able to adapt to different scenarios with varying levels of interaction and still be practically as safe as *full communication* policies.

However, it could be unclear whether the same results could still be achieved by finding the right weighting trade-off between collision event and communication event penalizations. To verify this, we modify the reward function by adding a weighting variable ρ , as shown in Equation 12, and attempt to fine-tune it by training three models under different values for it: $\rho = \{0.98, 0.90, 0.50\}$. Informed constant velocity estimations are enabled during training. Note that $\rho = 0.5$ results in the original reward function proposed in section 4.1.4.



(a) Collision rate



(b) Number of communication requests

Fig. 9: Performance evaluation over the ablated versions trained while enabling the prior information predictions. We add the performance of our own method for comparison. Standard deviations are taken over the results taken when evaluating 3 seeds of each trained policy over 1000 episodes.

$$R_i(\mathbf{x}^t, \pi_i^t) = w_g R_{g,i}(\mathbf{x}^t) + 2\rho w_{coll} R_{coll,i}(\mathbf{x}^t) + 2(1 - \rho) w_c R_{c,i}(\pi_i^t) \quad (12)$$

The values of ρ were chosen to showcase how difficult and counter-intuitive it is to properly balance communication and collision penalties when the training and test environment are the same. Rather than a wide range of parameters, Figure 9 shows which values of ρ are necessary to obtain

a policy that matches ours in terms of collision rates ($\rho = 0.98$), communication requests ($\rho = 0.90$), and what happens when we balance both objectives equally ($\rho = 0.50$).

In Figure 9, we compare our method with different versions of the ablated model trained under the given different values for ρ . Increasing the value of the ρ hyperparameter results in learning more conservative communication policies that make more communication requests to navigate more safely. We can argue that fixing such value around $\rho = 0.90$ yields similar performance to our method as it is just slightly lower in terms of communication requests in the most complex training scenario: *Asymmetric swapping*. However, we show crucial differences in terms of fine-tuning difficulty, adaptability and reliability of the learned policy. In rotation scenarios, we should learn to decrease communications as no interaction is needed to perform safe navigation in this setting.

Note how the value of ρ has high impact on the converged policy for the ablated versions. In particular, their collision rate and overall communication amount throughout all scenarios variate greatly as shown in Figure 9. In contrast, our method allows us to decrease the number of communications while hardly compromising safety by just applying a simple modification.

Additionally, ablated versions fail to adapt to different scenarios. Policies trained with high values for ρ (≥ 0.90) tend to over-communicate, requesting other robots' trajectory intentions even when both of them are not moving. Instead, for lower values of ρ (≤ 0.90 and specially ≤ 0.50), learned policies tend to under-communicate as they rely too much on the predicted trajectory intentions which compromises their collision rate. In particular, a balanced value of $\rho = 0.5$ that equally punishes collisions and communication already results in close to 0 communication requests and high collision rates. We won't get any further interesting results from lower ρ values since that would mean a policy even closer to the *no communication* policy baseline. Enabling informed constant velocity estimations during training results in learned policies that leverage how much they can rely on informed constant velocity estimations. In practice, this means that we have a stochastic policy that controls the expected frequency of trajectory intention

requests. While it is another valid strategy, it lacks adaptability and is less reliable and intuitive in complex scenarios. This explains why policies trained under high values for ρ tend to overcommunicate in all scenarios (Figure 9).

6 Real Experiments

In this section, we demonstrate that our communication policy learned through reinforcement learning in simulation can be deployed on physical quadrotors (see also Extension 2). In the following subsections, we first briefly introduce the hardware setup of our framework. Then, we present the multi-robot scenarios used for evaluation.

6.1 Hardware Setup

As in [Zhu and Alonso-Mora \(2019b\)](#), our experimental platform is the Parrot Bebop 2 quadrotor. The radius of each quadrotor is set as 0.30m in the MPC. An external motion capture system (Optitrack) is used to measure the pose of each quadrotor, which provides an estimated pose for each quadrotor. We then use an UKF to estimate the state of quadrotors ([Zhu & Alonso-Mora, 2019b](#)). We use an Intel i7 CPU@2.6GHz computer for the communication policy and planner and use Robot Operating System (ROS) to send commands to the quadrotors. The communication policy and the NMPC configurations are explained in section 5.1.

6.2 Multi-Robot Scenarios

In this section, we design three scenarios to validate that the behaviors learnt in simulation during training (Figure 3) can be reproduced in real multi-drone teams. These experiments have been designed to showcase the adaptability of the communication policy to different amount of drones and to motion planning tasks requiring different amounts of interaction.

First, in Figure 10a we let two drones follow parallel trajectories (analogous to the rotation scenario in 2-drone settings) to verify that the learned communication policy does not communicate when it is not necessary. Additionally, we let them swap positions (Figure 10b) to demonstrate that two robots can reliably avoid each

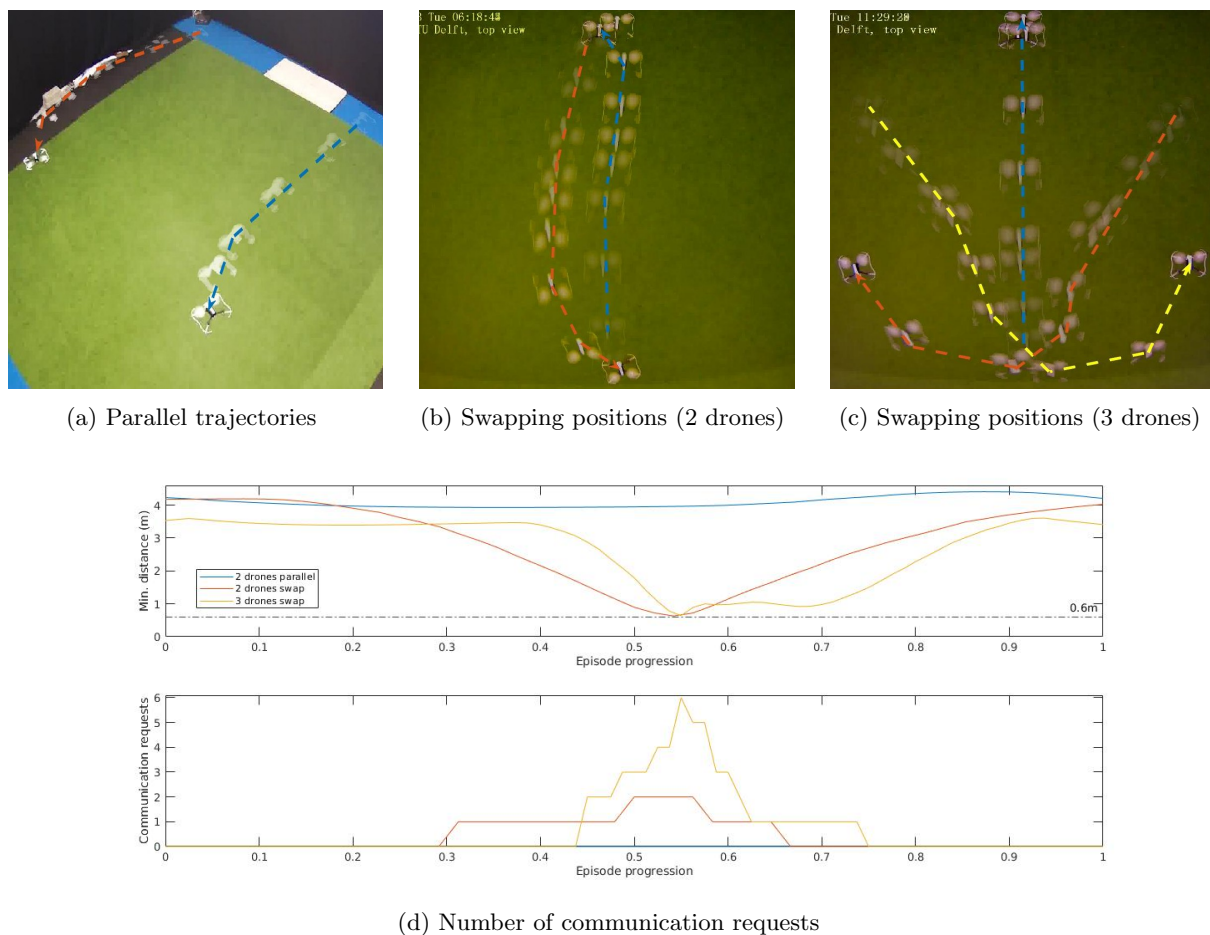


Fig. 10: Validation of our trained policy in real experiments.

other using this framework and to verify the adaptability of our learned communication policy to different situations along the episode (i.e. they do not communicate unless needed to avoid collisions). Finally, we add a third robot to the environment and let them perform the swapping scenario (Figure 10c). Note that the communication policy is the same for all robots and does not need retraining when their number changes.

In Figure 10d, we plot the minimum distance among drones and the number of communications registered along these three scenarios. We show that in all three cases, our framework manages to avoid collisions with a minimal number of communication requests, and to adapt to a different number of robots without retraining or tuning

the parameters from the NMPC.

To show the relationship between communication requests and distance holds even in real environments, we perform 9 swapping experiments with two drones (Figure 10b) while keeping record of the distance between them and the number of communications. Although there are overlaps among the distance distributions, the box plot in Figure 11 shows a clear relationship between the distance between drones and the number of communications. Note that the outliers in the 0-communication distribution and the overlaps between boxes could be due to the fact that the learned communication policy does not behave symmetrically in space. As seen in Figure 11, this means that the two drones do not necessarily communicate at the same time and does not behave

equally before and after the intersection (Figure 10d). Our results show therefore that the proposed learned communication strategy allows physical quadrotors to navigate tight situations with lower communication requests to avoid collisions.

7 Conclusions

In this paper, we have introduced an efficient communication policy integrating the strengths of MARL and NMPC in collision avoidance tasks. Simulation results show that our policy learns *when* and from *whom* to request planned trajectories to successfully avoid collisions. Experimental results show that the learned communication policy can be deployed on physical quadrotors. Further testing and the extension of our method to heterogeneous multi-robot systems is left for future work. Our method reduces the amount of communication requests significantly while achieving collision-free motions, practically achieving the same safety as more conservative communication baselines. The analysis and extension of our method under imperfect and delayed communication conditions are also left for future work. In comparison with Serra-Gómez et al. (2020), we use an architecture that enables us to scale our approach to higher and varying number of agents during and after training. Furthermore, we introduce a training method which allows to learn safe policies without sacrificing adaptability. Future work will investigate how to prioritize episodes from scenarios which are rich in information to improve sample efficiency. Finally, our learned communication policy can only influence and coordinate the motion planning of each robots to a certain extent. It can only choose when additional information is needed to generate safe trajectories, but cannot modify this information nor modify the plans generated by the NMPC directly. Learning how to modify the information and/or plans generated by the NMPC to compensate for a lack in accuracy of our model is left for future work as well.

Supplementary information. Accompanying this article we provide electronic supplementary material of our experiments in both simulated and real environments.

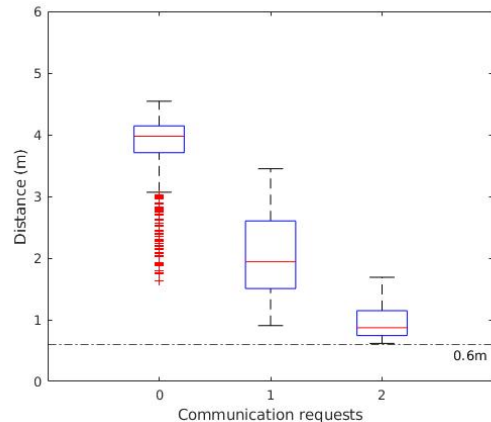


Fig. 11: Box plot on the two-drone distance distribution for different levels of communication. The dotted line indicates the collision distance.

Appendix A Dynamic model

In this work, we use the same drone model and specifications for the Parrot Bebop2 SDK as in Zhu and Alonso-Mora (2019b). According to the Parrot Bebop2 SDK, the control inputs to the quadrotor are given by $\mathbf{u} = [\phi_c, \theta_c, v_{z_c}, \dot{\psi}_c] \in \mathbb{R}^4$, where ϕ_c and θ_c are the desired roll and pitch angles, v_{z_c} is the desired linear velocity in the z-axis and $\dot{\psi}_c$ is the yaw rate. To simulate the drone dynamics, we extend the state of each drone, as defined in Section 4.1.1, with information of its orientation (ϕ, θ, ψ) . We use a first order low-pass Euler approximation of the quadrotor dynamics (Zhu & Alonso-Mora, 2019b), where the dynamics of the state velocity vector are:

$$\begin{cases} \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \end{bmatrix} = R_Z(\psi) \begin{bmatrix} \tan \theta \\ -\tan \phi \end{bmatrix} g - k_D \begin{bmatrix} v_x \\ v_y \end{bmatrix}, \\ \dot{v}_z = \frac{1}{\tau_{v_z}} (k_{v_z} v_{z_c} - v_z), \end{cases} \quad (\text{A1})$$

where $g = 9.81m/s^2$ is the earth's gravity, $R_Z(\psi) \in SO(2)$ is the rotation matrix along the drone's local z-axis, k_D is the drag coefficient, k_{v_z} and τ_{v_z} are the gain and time constant of vertical velocity. The attitude dynamics of the quadrotor are:

$$\dot{\phi} = \frac{1}{\tau_\phi} (\phi_c - \phi), \quad \dot{\theta} = \frac{1}{\tau_\theta} (\theta_c - \theta), \quad \dot{\psi} = \dot{\psi}_c \quad (\text{A2})$$

where τ_ϕ, τ_θ are the time constants of roll and pitch angles respectively. In this work, due to the drone being able to move in any direction with any yaw angle, we fix the drone's yaw angle to zero. Consequently, $\dot{\psi}_c = 0$

Appendix B Cost Functions

The components of the cost function $J_i^k(\mathbf{x}_i^k, \mathbf{u}_i^k), k = 0, 1, \dots, N - 1$ and $J_i^N(\mathbf{x}_i^N, \mathbf{g}_i)$ are defined in the following.

Goal navigation

We minimize the displacement between the trajectory's terminal position and the robot's goal location, and define a terminal cost

$$J_i^N(\mathbf{x}_i^N, \mathbf{g}_i) = w_i^N \frac{\|\mathbf{p}_i^N - \mathbf{g}_i\|}{\|\mathbf{p}_i^0 - \mathbf{g}_i\|}, \quad (\text{B3})$$

where $w_i^N \in \mathbb{R}^+$ is a tuning weight coefficient.

Control input cost

One of the stage cost terms is to minimize the control input,

$$J_{i,u}^k(\mathbf{u}_i^k) = w_{i,u} \|\mathbf{u}_i^k\|, \quad (\text{B4})$$

where $w_{i,u} \in \mathbb{R}^+$ is a tuning weight coefficient.

Collision cost

To improve safety, a stage collision potential field cost is introduced between the robot and each other robot,

$$J_{i,j,c}^k(\mathbf{x}_i^k) = \begin{cases} w_{i,c}(d^{\text{pot}} - d_{ij}^k), & \text{if } d_{ij}^k < d^{\text{pot}}, \\ 0, & \text{otherwise.} \end{cases} \quad (\text{B5})$$

where $w_{i,c} \in \mathbb{R}^+$ is a tuning weight coefficient, $d_{ij}^k = \|\mathbf{p}_i^k - \hat{\mathbf{p}}_j^k\|$ is the distance between robot i and each other robot j , and d^{pot} is the specified potential field distance, a scalar hyperparameter that establishes the limits of the potential field. The field grows linearly once a robot enters its limits as seen in equation B5. Then the collision potential cost is defined as

$$J_{i,c}^k(\mathbf{x}_i^k) = \sum_{j \in \mathcal{L}, j \neq i} J_{i,j,c}^k(\mathbf{x}_i^k). \quad (\text{B6})$$

Finally, the overall stage cost is

$$J_i^k(\mathbf{x}_i^k, \mathbf{u}_i^k) = J_{i,u}^k(\mathbf{u}_i^k) + J_{i,c}^k(\mathbf{x}_i^k). \quad (\text{B7})$$

Acknowledgements. This work is supported by the U.S. Office of Naval Research Global (ONRG) NICOP-grant N62909-19-1-2027.

Declarations

Conflict of interest. The authors declare that they have no conflict of interest.

Author contribution. All authors contributed to the study conception and design. Álvaro Serra-Gómez wrote the first draft of the manuscript, implemented the method and simulated experiments, and analysed the obtained results. Hai Zhu and Bruno Brito wrote the first draft of related works section 2.1 and participated in writing the preliminaries section. Álvaro Serra-Gómez and Hai Zhu implemented the simulation environment used for training and testing. Álvaro Serra-Gómez, Hai Zhu and Bruno Brito performed the real experiments. Wendelin Böhmer provided supervision and feedback over the whole project, with a specific focus on the method and analysis of results. Javier Alonso-Mora supervised and provided feedback for the whole project, including the analysis of results, and provided funding. All authors commented on previous versions of the manuscript. All authors reviewed and approved the final manuscript.

References

- Becker, R., Carlin, A., Lesser, V., Zilberstein, S. (2009). Analyzing myopic approaches for multi-agent communication. *Computational Intelligence*, 25, 31–50. 10.1111/j.1467-8640.2008.01329.x
- Bengio, Y., Louradour, J., Collobert, R., Weston, J. (2009). Curriculum learning. *Proceedings of the 26th annual international conference on machine learning* (pp. 41–48).
- Bernstein, D., Givan, R., Immerman, N., Zilberstein, S. (2002, 12). The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27. 10.1287/moor.27.4.819.297

- Best, G., Forrai, M., Mettu, R.R., Fitch, R. (2018). Planning-aware communication for decentralised multi-robot coordination. *2018 IEEE International Conference on Robotics and Automation (ICRA)* (p. 1050-1057). 10.1109/ICRA.2018.8460617
- Brito, B., Everett, M., How, J.P., Alonso-Mora, J. (2021). Where to go next: Learning a subgoal recommendation policy for navigation among pedestrians. *ArXiv, abs/2102.13073*.
- Das, A., Gervet, T., Romoff, J., Batra, D., Parikh, D., Rabbat, M., Pineau, J. (2019). TARMAC: Targeted multi-agent communication. *36th International Conference on Machine Learning, ICML 2019*.
- Ding, Z., Huang, T., Lu, Z. (2020). Learning individually inferred communication for multi-agent cooperation. *ArXiv, abs/2006.06455*.
- Domahidi, A., & Jerez, J. (2014). *Forces professional*. embotech gmbh (<http://embotech.com/forces-pro>).
- Everett, M., Chen, Y.F., How, J.P. (2018). Motion Planning among Dynamic, Decision-Making Agents with Deep Reinforcement Learning. *Ieee International Conference on Intelligent Robots and Systems* (pp. 3052–3059). Retrieved from https://github.com/mfe7/cadrl_ros 10.1109/IROS.2018.8593871
- Everett, M., Chen, Y.F., P. How, J. (2019). Collision avoidance in pedestrian-rich environments with deep reinforcement learning. *ArXiv, abs/1910.11689*.
- Fan, T., Long, P., Liu, W., Pan, J. (2020). Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *The International Journal of Robotics Research*, 39(7), 856-892. Retrieved from <https://doi.org/10.1177/0278364920916531> 10.1177/0278364920916531
- Foerster, J.N., Assael, Y.M., de Freitas, N., Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. *Proceedings of the 30th international conference on neural information processing systems* (p. 2145–2153). Red Hook, NY, USA: Curran Associates Inc.
- Foerster, J.N., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S. (2018). Counterfactual multi-agent policy gradients. *Proceedings of the thirty-second AAAI conference on artificial intelligence and thirtieth innovative applications of artificial intelligence conference and eighth AAAI symposium on educational advances in artificial intelligence*. AAAI Press.
- Gama, F., Marques, A., Leus, G., Ribeiro, A. (2019). Convolutional neural network architectures for signals supported on graphs. *IEEE Transactions on Signal Processing*, 67, 1034-1049.
- Gupta, J.K., Egorov, M., Kochenderfer, M. (2017). *Cooperative Multi-agent Control Using Deep Reinforcement Learning* (Vol. 10642 LNAI; Tech. Rep.). 10.1007/978-3-319-71682-4_5
- Hochreiter, S., & Schmidhuber, J. (1997, 12). Long short-term memory. *Neural computation*, 9, 1735-80. 10.1162/neco.1997.9.8.1735
- Iqbal, S., & Sha, F. (2019). Actor-attention-critic for multi-agent reinforcement learning. *ICML*.
- Jiang, J., & Lu, Z. (2018). Learning attentional communication for multi-agent cooperation. *Advances in neural information processing systems*.
- Kamel, M., Alonso-Mora, J., Siegwart, R., Nieto, J. (2017). Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 236–243). IEEE. 10.1109/IROS.2017.8202163
- Kassir, A., Fitch, R., Sukkarieh, S. (2016, nov).

- Communication-efficient motion coordination and data fusion in information gathering teams. *Ieee international conference on intelligent robots and systems* (Vol. 2016-November, pp. 5258–5265). Institute of Electrical and Electronics Engineers Inc. 10.1109/IROS.2016.7759773
- Kingma, D.P., & Ba, J. (2015). Adam: A method for stochastic optimization. *CoRR*, *abs/1412.6980*.
- Kurin, V., Igl, M., Rocktäschel, T., Boehmer, W., Whiteson, S. (2020). My body is a cage: the role of morphology in graph-based incompatible control. *ArXiv*, *abs/2010.01856*.
- Li, Q., Gama, F., Ribeiro, A., Prorok, A. (2020). Graph neural networks for decentralized multi-robot path planning. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 11785-11792.
- Li, Q., Lin, W., Liu, Z., Prorok, A. (2020). Message-aware graph attention networks for large-scale multi-robot path planning. *ArXiv*, *abs/2011.13219*.
- Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Gonzalez, J., Goldberg, K., Stoica, I. (2017). Ray rllib: A composable and scalable reinforcement learning library. *ArXiv*, *abs/1712.09381*.
- Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems* (Vol. 2017-Decem, pp. 6380–6391).
- Luis, C.E., Vukosavljev, M., Schoellig, A.P. (2020). Online trajectory generation with distributed model predictive control for multi-robot motion planning. *IEEE Robotics and Automation Letters*, 5(2), 604–611. 10.1109/LRA.2020.2964159
- Mordatch, I., & Abbeel, P. (2018). Emergence of grounded compositional language in multi-agent populations. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 1495–1502.
- Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M.I., Stoica, I. (2018). Ray: A distributed framework for emerging ai applications. *Proceedings of the 13th usenix conference on operating systems design and implementation* (p. 561–577). USA: USENIX Association.
- Rahmattalabi, A., Chung, J.J., Colby, M., Tumer, K. (2016). D++: Structural credit assignment in tightly coupled multiagent domains. *2016 ieee/rsj international conference on intelligent robots and systems (iros)* (p. 4424-4429). 10.1109/IROS.2016.7759651
- Rashid, T., Samvelyan, M., Witt, C.S.D., Farquhar, G., Foerster, J.N., Whiteson, S. (2018). Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *ArXiv*, *abs/1803.11485*.
- Roth, M., Simmons, R., Veloso, M. (2005). *Reasoning about joint beliefs for execution-time communication decisions* (Tech. Rep.).
- Schulman, J., Moritz, P., Levine, S., Jordan, M.I., Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. *CoRR*, *abs/1506.02438*.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017). Proximal policy optimization algorithms. *ArXiv*, *abs/1707.06347*.
- Serra-Gómez, A., Brito, B., Zhu, H., Chung, J.J., Alonso-Mora, J. (2020). With whom to communicate: Learning efficient communication for multi-robot collision avoidance. *2020 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 11770–11776). IEEE.
- Son, K., Kim, D., Kang, W.J., Hostallero, D.E., Yi, Y. (2019). Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. *ArXiv*,

abs/1905.05408.

- Sukhbaatar, S., Szlam, A., Fergus, R. (2016). Learning multiagent communication with backpropagation. *Advances in Neural Information Processing Systems*(Nips), 2252–2260.
- Sun, C., Shen, M., How, J.P. (2020). Scaling up multiagent reinforcement learning for robotic systems: Learn an adaptive sparse communication graph. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 11755-11762.
- Sunehag, P., Lever, G., Gruslyns, A., Czarnecki, W., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J.Z., Tuyls, K., Graepel, T. (2018). Value-decomposition networks for cooperative multi-agent learning. *ArXiv, abs/1706.05296*.
- Talamali, M.S., Saha, A., Marshall, J.A.R., Reina, A. (2021). When less is more: Robot swarms adapt better to changes with constrained communication. *Science Robotics*, 6.
- Van Den Berg, J., Guy, S.J., Lin, M., Manocha, D. (2011). Reciprocal n-body collision avoidance. *Springer tracts in advanced robotics* (Vol. 70, pp. 3–19). 10.1007/978-3-642-19457-3_1
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. (2017). Attention is all you need. *ArXiv, abs/1706.03762*.
- Wang, L., Ames, A.D., Egerstedt, M. (2017). Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics*, 33(3), 661–674. 10.1109/TRO.2017.2659727
- Wang, R.E., Kew, J., Lee, D., Lee, T., Zhang, T., Ichter, B., Tan, J., Faust, A. (2020). Model-based reinforcement learning for decentralized multiagent rendezvous. *arXiv: Multiagent Systems*.
- Wheeler, T., Bharathi, E., Gil, S. (2019). Switching topology for resilient consensus using wi-fi signals. *2019 international conference on robotics and automation (icra)* (p. 2018-2024). 10.1109/ICRA.2019.8793788
- Yongjie, Y., & Yan, Z. (2009). Collision avoidance planning in multi-robot based on improved artificial potential field and rules. *2009 IEEE international conference on robotics and biomimetics (robio)* (pp. 1026–1031). IEEE.
- Zhai, Y., Ding, B., Liu, X., Jia, H., Zhao, Y., Luo, J. (2021). Decentralized multi-robot collision avoidance in complex scenarios with selective communication. *IEEE Robotics and Automation Letters*, 6(4), 8379-8386. 10.1109/LRA.2021.3102636
- Zhou, D., Wang, Z., Bandyopadhyay, S., Schwager, M. (2017). Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells. *IEEE Robotics and Automation Letters*, 2(2), 1047–1054. 10.1109/LRA.2017.2656241
- Zhu, H., & Alonso-Mora, J. (2019a). B-uavc: Buffered uncertainty-aware voronoi cells for probabilistic multi-robot collision avoidance. *2019 international symposium on multi-robot and multi-agent systems (mrs)* (pp. 162–168).
- Zhu, H., & Alonso-Mora, J. (2019b). Chance-constrained collision avoidance for mavs in dynamic environments. *IEEE Robotics and Automation Letters*, 4(2), 776–783. 10.1109/LRA.2019.2893494
- Zhu, H., Claramunt, F.M., Brito, B., Alonso-Mora, J. (2021). Learning interaction-aware trajectory predictions for decentralized multi-robot motion planning in dynamic environments. *IEEE Robotics and Automation Letters*, 6(2), 2256-2263. 10.1109/LRA.2021.3061073
- Zhu, H., Juhl, J., Ferranti, L., Alonso-Mora, J. (2019). Distributed multi-robot formation splitting and merging in dynamic environments. *2019 IEEE international conference on*

robotics and automation (icra) (pp. 9080–9086). IEEE. 10.1109/ICRA.2019.8793765