

Multi-agent deep reinforcement learning in mobile robotics

Rapport du comité de suivi de 1ère année

Maxime Toquebiau

ECE Paris, Research Center, 37 quai de Grenelle, 75015, Paris, France

Sorbonne Université, Institut des Systèmes Intelligents et de Robotique, ISIR,
4 place Jussieu, 75005 Paris, France

February 2022

Jury:

Mme. Aurélie Beynier , LIP6	Rapporteur
M. Olivier Simonin , INSA Lyon	Rapporteur
M. Nicolas Bredeche , ISIR	Co-directeur
M. Faïz Benamar , ISIR	Co-directeur
M. Jae-Yun Jun , ECE Paris	Encadrant

This document presents the current status of my PhD thesis "Multi-agent deep reinforcement learning in mobile robotics" which started in January 2021. In this document, I will start by presenting myself, my supervisors and how we work together. The subject of this thesis will then be explained, with its context and the main issues that will be treated. Next, the research directions we chose to pursue will be presented. Then, I will present the work done during this first year of my PhD, show some preliminary results and explain what will be the next steps. Finally, I will describe what training courses I have chosen to follow as part of my mandatory training plan.

1 Presentation of the research project

Recently, deep reinforcement learning has started to be successfully applied to Multi-Agent Systems (MAS), showing the capacity to learn coordinated behaviours in teams of cooperative agents ([Lowe et al., 2017](#); [Rashid et al., 2018](#); [Berner et al., 2019](#)). Ultimately, the goal is to use these algorithms in multi-robot systems (MRS) such as self-driving cars or swarms of drones. However, applying DRL to real-world robotics is still an on-going issue ([Ibarz et al., 2021](#)) and multi-agent deep reinforcement learning (MADRL) research focuses on toy environments such as sequential social dilemmas or video games ([Rashid et al., 2018](#); [Foerster et al., 2018](#); [Mahajan et al., 2019](#); [Son et al., 2019](#); [Berner et al., 2019](#); [Rashid et al., 2020](#); [Schroeder de Witt et al., 2020a](#)). While these environments capture some aspects of multi-agent interactions, they often lack some attributes of robotic tasks like partial observability or non-stationarity of the environment. To apply MADRL in mobile robotics, we need first to ask ourselves: What does having agents living in the real world imply? Answering this question will help us design simulated tasks that are more relevant to robotics than the ones studied in MADRL research today.

In future tasks, robots will have to comply to pre-defined rules, some of which may be rather abstract (e.g. social rules). They will need to interact with numerous heterogeneous agents: other kinds of robots, human-driven machines and humans. They will have to adapt to changes in their environment, with new agents coming into play or modifications of their working environment. Finally, they will have to communicate with each other and with humans, using

interpretable communication protocols (e.g., natural language). They will have to be able not only to communicate what they see and what they are doing, but also what they need. In the same way, they will need to understand the needs of their peers and humans they interact with. Our goal will be to make a few steps in this direction using the great potential of MADRL models.

2 Research directions

After defining the context of this project, we have to define what questions we want to focus on during this thesis. The goal is to participate in the effort of applying MADRL to mobile robots, keeping in mind all the issues mentioned above. To guide our work toward this goal, we chose to pursue four different research directions that we will define here.

2.1 Emergent communication in MADRL

In an environment with *partial observability* and *non-stationarity* due to multiple agents concurrently learning, communication is a crucial tool for agents to be efficient. When doing decentralised execution, communication is the only way to entirely leverage the information gathered by all the agents in our system. Despite this clear potential, communication is still not used enough in the MADRL literature. At execution time, agents usually rely only on their observation history to choose their action and try to coordinate with other agents (Lowe et al., 2017; Rashid et al., 2018; Foerster et al., 2018; Schroeder de Witt et al., 2020a). This is unfortunate as communication, when done correctly, can be beneficial to decentralised MAS (Tan, 1993). Thus, this first motivates a study of communication systems in MAS and their impact on MADRL approaches. State-of-the-art MADRL algorithms (Rashid et al., 2018; Foerster et al., 2018; Schroeder de Witt et al., 2020a,b; Wang et al., 2021) can be augmented with an *emergent communication system* (Foerster et al., 2016; Sukhbaatar et al., 2016; Mordatch and Abbeel, 2018; Jiang and Lu, 2018; Das et al., 2019) to see how communication can improve the performance in this task. In emergent communication, agents create their own language to convey information about their observations and their strategy. This language, usually made of continuous signals, is created while the agents are learning to perform their task. Such communication systems can be very helpful for sharing local information unavailable to other agents and coordinate actions. Our contribution will be to study how learning an emergent communication system impacts performance of MADRL algorithms. Specifically, how emergent communication can help alleviate common issues of these algorithms such as relative over-generalisation (Wei and Luke, 2016), and how it can help independent learners to equal or surpass centralised learners (Schroeder de Witt et al., 2020a).

2.2 Learning an existing language

Instead of creating a new language during training, it might be more valuable to learn a pre-existing discrete communication system like natural language. It is still very hard to interpret emergent languages learnt by teams of agents, or even to measure their impact on the agents' policy (Kottur et al., 2017; Lowe et al., 2019; Lazaridou and Baroni, 2020). But, as we said, robots will need to communicate with human beings with an interpretable language. This first means that the language that we use must be discrete, just like natural language is composed of discrete tokens (i.e., words and symbols). Second, this means that the language already exists beforehand, unlike in Section 2.1 where the language emerges during training. The tokens all have pre-defined meanings and must be arranged into sentences following strict grammatical rules. This adds a layer of complexity, but it is required if we want to develop systems that we can interact fully with. Also, ideas from developmental psychology, pioneered by soviet psychologist Vygotsky (1934), state that language is not just a tool for communication, but also plays a part in the cognitive development of humans. Words and phrases, arranged in sentences, are ways for us to understand the world and construct our thoughts. This has led to a whole body of research on language-augmented reinforcement learning (LA-RL) that uses language to guide agents during their task by defining goals (Andreas et al., 2017; Das et al., 2018; Colas et al., 2020; Akakzia et al., 2021). This shows that learning a pre-existing, discrete language can not only help us communicate in a interpretable way, but also drive the learning of multiple agents.

Learning an existing language is hard, even for humans. Traditional natural language methods usually rely on extensive statistical analysis of big corpora of text. This approach lacks of the

interactiveness that we would like in a intelligent agent learning to perform a task. In the reinforcement learning (RL) literature, agents have been learning natural language in simple communication games (Lazaridou et al., 2016; Das et al., 2017) and in embodied tasks in previously cited LA-RL approaches. But an existing language has never been learnt in a complex multi-agent environment. The challenge will be to induce the agents to learn to use the language properly, allowing them to improve their performance. This will most likely require a combination of interaction and supervision (Lowe et al., 2020). Agents will interact with one another and discover good ways to use the language. This could be enforced through reward shaping, for example by penalising bad grammar and rewarding good use of words. Supervision can be done in multiple ways. One option is to draw inspiration from LA-RL approaches by providing descriptions to agents about their observations and actions, from which they will extract some meaning about the words (Andreas et al., 2018; Nguyen et al., 2021; Karch et al., 2021). It could also be possible to perform imitation learning and make the agents learn from demonstrations of pre-programmed agents that use the language in the way we intended (Hester et al., 2018).

For this approach to be achievable, a simple and efficient language needs to be defined. Teaching natural language to robots would be a great breakthrough, but in the current context this is way out of reach. Indeed, human languages are made to convey information in millions of different situations. In comparison, our task is extremely simple. Therefore, the language we use must be simple as well (Mordatch and Abbeel, 2018; Lazaridou and Baroni, 2020). Table 1 presents a set of tokens that could compose our language. To read and generate such tokens, techniques from natural language processing such as language models could be used. This is a restricted list, but other tokens can be added according to what we want to incorporate in our language. For example, tokens could be added to express orders that, if available only to some agents, would induce social rules in our system. If this approach is successful, it would be a first step towards learning an interpretable language in a robotic task.

Type	Token
Entities	AGENT PACKAGE
Locations	NORTH SOUTH EAST WEST DELIVERY_AREAL
Verbs	GOING_TO NEED_HELP PUSH_PACKAGE

Table 1: Possible tokens used in a language designed for a delivery task.

2.3 Multi-agent credit assignment

In a cooperative scenario, agents have to learn to maximise a global team reward. However, as we have said in previous sections, agents have to face both partially observability and non-stationary. These two properties make the global reward unsuitable for agents to learn their optimal policy, as they are not able to deduce the quality of their actions from the reward obtained at a team level (Chang et al., 2004). This essentially means that we are faced with a *multi-agent credit assignment* problem where we need to find a way to know how much each agent’s actions have participated in the team reward. Multiple techniques have tackled the multi-agent credit assignment problem. For low-dimensional environments, optimal approaches have been developed, like the *shapley value* (Shapley, 1953) or the *wonderful life* and *aristocrat* utilities (Wolpert and Tumer, 2002). More recently, these methods have been adapted to larger environments with the use of DRL techniques (Foerster et al., 2018; Nguyen et al., 2018; Zhou et al., 2020). We need to study these solutions in our context and try to find new ones that yield better results. Specifically, we will have to link credit assignment to communication. Communication actions must impact the behaviour of other agents and improve the final global reward. Thus, our credit assignment strategy must be designed in relation to the communication system, which has never been studied before.

2.4 Macro-actions as a sim2real approach

Perhaps the most important issue when dealing with DRL and robotics is how to use algorithms trained in simulation in the real world. DRL methods are notoriously hard to apply in the real world (Sünderhauf et al., 2018; Ibarz et al., 2021). We need to find ways to effectively do it, otherwise we will never be able to leverage the power of DRL in robotics. The main issue here is that DRL is highly sample inefficient: it needs to train on great numbers of steps to converge to a good policy. This makes it impractical to train in reality, as experiences in the real world

take a long time to execute, way more than in simulation. In addition to that, RL involves doing numerous series of bad actions in order to learn not to execute them in the future, leading to potentially very bad outcomes for the agents. This is obviously undesirable with real robots, as they are usually quite brittle and very expensive.

One solution to these issues is to use the data gathered in simulation to learn behaviours and try to transfer them on real robots. This is challenging due to the multiple differences between simulated environments and reality, all summarised as the '*reality gap*'. This has led to numerous simulation-to-reality (sim2real) approaches that use different tricks to bridge this gap (Tobin et al., 2017; Peng et al., 2018; James et al., 2019; Chebotar et al., 2019; Andrychowicz et al., 2020).

Here, we describe how macro-actions could be used as a sim2real approach. Macro-actions are temporally-extended actions that execute multiple low-level actions. Based on the options framework of Sutton et al. (1999), they were used by Amato et al. (2019) and Xiao et al. (2020) to learn efficient robotic policies. They allow us to leverage pre-programmed controllers for solving relatively easy sub-problems like navigating to a location. Navigating from one point to another in a known environment is a task that can be completed rather easily. Using RL to learn such tasks with robots can be extremely complicated and time consuming. It is thus more pertinent to focus the learning on the more complex part of the policy we want to learn: when to go where, how to coordinate and distribute sub-tasks. To this end, macro-actions can be used, such as "*go to zone X*" executing a pre-programmed policy to perform this macro-action. The agents would then learn a macro-policy that maximises their reward. Now, considering that we have pre-designed controllers for executing all of our macro-actions in the real-world, we could transfer the macro-policy learnt in a simulated environment to a real environment with the same layout. The macro-policy being more high-level, it is less affected by the difference in executing its actions between simulation and reality.

For this approach to be possible, we will have to face multiple challenges. First, we need to address the differences in the inputs of our agents between simulation and reality. If we want to base our robots' observations only on their local sensors, then we will have to transfer the model from the simulated inputs to the sensor inputs. Second, even if macro-actions help us shortening the reality gap, differences in executing actions between simulation and reality may still impact the quality of the macro-policy. For example, if we have pre-designed controllers to go from point A to point B both in simulation and reality, these two controllers might perform this task in different fashions (e.g., with different path or speed). This might require some work on the simulation side to ensure that these differences do not impact the performance. In this regard, we can draw inspiration from domain and dynamics randomisation techniques in other sim2real approaches (Tobin et al., 2017; Peng et al., 2018; Andrychowicz et al., 2020). If we manage to overcome these difficulties, this has the potential to be one of the first sim2real approaches for multiple mobile robots.

3 Current work

In the beginning, the subject definition of this thesis was very broad. It embraced every aspects of the three main themes of DRL, MAS and mobile robotics, without specifying a particular issue to focus on. A lot of work has then been dedicated to finding what were the main questions we wanted to work on. In this section, we will present all the successive steps taken this year from literature review to training existing methods, and finally talk about what we will focus on in the future.

3.1 Review of the literature

Starting working on this thesis, the first step was to review the existing literature. The goal was to familiarise myself with our domains of interest in order to find what issues I wanted to address. Thus, before focusing on specific subjects, the first task was to understand the main concepts used in DRL and MAS. With the appropriate background knowledge, it would then be possible to study recent successful approaches in MADRL and how they were used in mobile robotics.

3.1.1 Deep Reinforcement Learning

The first domain we looked into was Deep Reinforcement Learning (DRL). Before talking about DRL, let us quickly define reinforcement learning. Reinforcement learning is a machine learn-

ing approach that allows artificial agents to learn the optimal strategy to complete a task. They do so by trying different actions and receiving rewards, positive or negative, depending on the quality of the action. The optimal strategy in time step t is the one that maximises the return G_t : the sum of future discounted rewards r ,

$$G_t = \sum_{k=1}^{\infty} \gamma^k r_{t+k},$$

with $\gamma \in [0, 1]$ a discount factor that gives more importance to rewards closer in time. To learn this strategy, three different tools can be used:

- Policy-based approaches use a **policy function** π to predict the best action a to perform in a given state s :

$$\pi(s) = a.$$

- Value-based approaches use a **value function** to predict the quality of a state or an action in a given state. The quality, or value of a state is defined as the expected return starting from the current state s_t and following a given policy π :

$$V_{\pi}(s_t) = E_{\pi}[G_t|s_t].$$

In Q-learning, the value of an action, or action-value, in a given state is defined as

$$Q_{\pi}(a_t|s_t) = E_{\pi}[G_t|a_t, s_t].$$

- Model-based approaches use a **model of the environment** to predict the next states of the environment and use this prediction for planning ahead of time.

DRL is the application of deep neural networks (DNN) in reinforcement learning. It started during the last decade following the explosion of deep learning methods (Mnih et al., 2013) and quickly became the main focus of research in reinforcement learning. Deep learning helps solving scalability and generalisation issues of classical reinforcement learning methods. There are multiple ways of using DNN in reinforcement learning, we can classify all approaches in three different groups: value-based, policy-based and model-based approaches.

Value-based methods use a DNN to compute the value function. The network takes as input the current state and outputs either the value of this state, or the action-values of each action in this state. The latter corresponds to a Deep Q-Network (DQN) (Mnih et al., 2013, 2015) which was the first successful DRL model. Using recent progress in deep learning, it managed to achieve professional human level in 49 two-dimensional video games, using only raw pixel images as input. In the next years, DQNs have been progressively improved to tackle various issues such as sample inefficiency (Schaul et al., 2016), strong variance in training (van Hasselt et al., 2016; Wang et al., 2016; Hessel et al., 2017), exploration (Fortunato et al., 2017) and partially observable environments (Kapturowski et al., 2018).

Policy-based methods also use DNNs as their policy function. An issue with DQNs is that they can only be used with discrete action domains. This can be solved using policy functions in place of or along with a DQN. Deep Deterministic Policy Gradient (DDPG) is one of these mixed approaches, called actor-critics, that combine policy and value functions (Lillicrap et al., 2015). The policy function computes a continuous action with a DNN and a DQN predicts its action-value. Other approaches of this kind have been developed like Proximal Policy Optimisation (PPO) to prevent unstable training using more precise gradient updates (Schulman et al., 2015, 2017). Also, DQN architecture improvements and other tricks like delayed policy updates and entropy have been used to make the actor-critic framework even more efficient in Twin Delayed DDPG (TD3) (Fujimoto et al., 2018) and Soft Actor-Critic (SAC) (Haarnoja et al., 2018). All these architectures are still considered state-of-the-art approaches and are widely used in research.

Of all approaches in DRL, the ones that were the most astounding were model-based methods. Quickly after the first breakthrough in DRL, DQNs were used along with classical model-based planning techniques to tackle classic games like Chess and Go. The latter had long been a major challenge for artificial intelligence research as it features an extremely large number of different states, and it can be hard to assess the quality of those states. This ended in 2015 when AlphaGo was released (Silver et al., 2016). Using a combination of value and policy networks with Monte Carlo Tree Search, they managed to beat professional players in the full game of GO. It was later improved with AlphaGo Zero that used only self-play for training (Silver et al.,

2017b) and AlphaZero that learnt faster and generalised to other games like Chess and Shogi (Silver et al., 2017a). Since then, DNNs have been used to improve the planning potential of model-based models by trying to predict different future outcomes using representation learning (Hafner et al., 2018, 2019) and recurrent world models (Ha and Schmidhuber, 2018; Schrittwieser et al., 2019).

All previously cited methods are the most influential of the past years and constitute the basis of almost all state-of-the-art approaches in DRL today. They all took basic principles of reinforcement learning and used DNN tools in clever ways to solve issues of generalisation, scalability and others. However, these methods still have some shortcomings and DRL research has also explored other strategies to overcome these. For instance, exploration of the environment is still a major issue. DRL methods usually employ passive exploration strategies to discover new states (e.g., taking random actions) which can be extremely inefficient, especially in environments with very sparse rewards. This issue led to more elaborate exploration strategies like intrinsic objectives to push agents exploring their environment (Pathak et al., 2017; Burda et al., 2019; Sekar et al., 2020) or actively looking for unexplored promising states (Shyam et al., 2019; Ecoffet et al., 2019, 2021). These active exploration strategies can help for more efficient training by finding more often states we want to attain. Thus, they could be helpful in a robotic environment where training agents is particularly expensive.

3.1.2 Multi-Agent Deep Reinforcement Learning

After reviewing state-of-the-art papers in DRL, next step was to look into the applications of DRL in Multi-Agent Systems (MAS). Having multiple agents concurrently learning inside an environment makes the learning process a lot more complex. We have already mentioned the two main issues with MAS that are partial observability and non-stationarity. The former is also bound to real-world robotic environments and refers to the fact that agents don't have the full state of the environment, but instead only an individual local observation. Therefore, agents have to predict the best action to perform with an incomplete knowledge about the state of the environment and, in a multi-agent system, the strategy of other agents. Non-stationarity describes the fact that because all agents are learning concurrently, the environment, from the point of view of an agent, is constantly evolving. Therefore, from the agent's perspective, the optimal policy it is trying to find is always changing. To these two major issues, we can add the simple fact that adding agents means increasing the dimension of the joint state and action spaces. We have more joint states to visit and more joint actions to try to hopefully find the best joint policy for our multi-agent system. All these issues make multi-agent environments difficult for learning. However, they are still of great interest as MAS can model many interesting situations. Real-world environments are composed of multiple various entities that can very much be considered as individual agents interacting. Thus, MAS help us bring artificial intelligence into these highly interactive environments.

To this end, Multi-Agent Deep Reinforcement Learning (MADRL) tries to leverage recent breakthrough in DRL to deal with issues of MAS. It does so by using different tools. One of the most important of these tools is Centralised Training and Decentralised Execution (CTDE) (Kraemer and Banerjee, 2016). The idea behind CTDE is that while we want to have decentralised agents that can take decisions without relying on a centralised decision process, we still can leverage some centralised information during training to converge more easily to the optimal joint policy. This paradigm fits well into the actor-critic approach, where the actor can be completely decentralised while the critic can use centralised information as it is used only during training. This is the approach of the Multi-Agent DDPG (MADDPG) model that adapts DDPG to the multi-agent case (Lowe et al., 2017). It defines each agent as a decentralised policy network that uses only the agent's local observation, coupled with a DQN that uses the joint state of all agents in the system as input. While CTDE is not the only approach possible it is the most promising, producing the best state-of-the-art models (Foerster et al., 2018; Rashid et al., 2018; Son et al., 2019). Most of the methods we will study next fit into this paradigm. Another possible way is independent learning (Wei and Luke, 2016; Tampuu et al., 2017; Schroeder de Witt et al., 2020a) which considers other agents in the system as part of the environment. This is computationally more efficient as we do not try to find the optimal joint policy but only the local one, but learning is harder due to non-stationarity and partial observability of the environment. Finally, centralised execution is also studied (Sukhbaatar et al., 2016; Böhmer et al., 2020), which simplifies the multi-agent problem as if there was only one process controlling all the agents. This makes learning and execution easier as we have more information on the global state of the environment, but it requires full centralisation at execution time which is often impractical.

Another important tool is credit assignment. As described in [Section 2.3](#), credit assignment is a way to divide the global reward between agents to match the quality of their actions. In MADRL, techniques have tried to adapt past ideas like the aristocrat utility to DRL ([Nguyen et al., 2018](#)). One important method is Counterfactual Multi-Agent (COMA) policy gradient ([Foster et al., 2018](#)). COMA uses a centralised critic to devise a counterfactual baseline used to compare the quality of the chosen action to an estimated average quality of other actions.

Credit assignment can also be done implicitly by learning value functions for each agent in the system, which is called value factorisation or decomposition. Indeed, as we have seen, the value function is an estimator of future returns. So if we have one action-value function for each agent, we would expect that these values reflect the actual quality of the corresponding agents' actions. To this end, we may use one decentralised critic for each agent and a way to reconstruct the total joint action-value of all the agents. Value-Decomposition Networks (VDN) does that by assuming the joint action-value is the sum of all the individual action-values ([Sunehag et al., 2018](#)). In QMIX, the individual action-values are mixed into the joint action-value using a separate DNN whose parameters depend on the current state of the environment ([Rashid et al., 2018](#)). These two methods have shown the great potential of value factorisation. VDN is still often took as baseline in research and QMIX is considered as one of the best state-of-the-art models. In addition, a lot of research has been done to extend QMIX in order to deal with its theoretical limitations ([Son et al., 2019](#); [Rashid et al., 2020](#); [Wang et al., 2021](#)) or with exploration ([Mahajan et al., 2019](#)).

In conclusion, there are multiple different approaches for MADRL. We will have to choose which are the best for our needs. For this matter, we might consider many factors. First, the number of parameters can vary a lot between models, which will have an impact on computing power required to train and run them. Also, some methods have core limitations, like value-based models that are limited to discrete action domains which can be an issue in robotics. Anyway, MADRL is a very young research domain, and models are usually designed for tool environments that try to replicate the dynamics of MAS. Depending on the task, models can have extremely uneven performance. For instance, value factorisation methods are prone to the issue of relative over-generalisation ([Wei and Luke, 2016](#)). It happens when the optimal joint-action is very hard to find as it is composed of individual actions that usually lead to bad outcomes. Value factorisation agents will thus converge to sub-optimal strategies where policy-based agents can have better results. Thus, we will certainly have to train and compare different solutions to see which fit our needs best.

3.1.3 Learning in mobile robotics

After reviewing the domains that would be at the technical core of this thesis, we needed then to understand the context of application in order to find research issues we would want to address. In our case, the domain in which we want to apply MADRL is mobile robotics. It refers to all robotic systems that can autonomously move inside their environment. This definition implies multiple specific constraints. First of all, our environment is dynamic. As a robot moves, its perspective of the environment changes. It also has to take into consideration other robots and moving parts of the environment. All of this makes planning more difficult. Second, if we work with robots, we need to take into account their sensors and actuators. These usually have imperfections that lead to unexpected behaviours. In addition to that, robotic environments can have various inherent constraints like varying weather conditions or limited communication. Finally, real-world environments are extremely complex. Moreover, reinforcement learning usually involves gathering a lot of bad experiences in order to learn the best strategy, which is incompatible with safety constraints and expensive equipment. This is why simulation is preferred for training models. However, whether we want to work in the real-world or in simulation, we have to take into consideration all previously mentioned constraints to design systems that are relevant to real-world applications.

All these issues make DRL impractical to apply in robotics. DRL models that struggle to learn an efficient strategy in simulation will struggle even more to learn or use this strategy in the real world. However, the great potential of these models motivates research to find solutions for applying DRL to robotics. Specifically, in mobile robotics, DRL has been used for autonomous navigation from raw pixel images ([Zhu et al., 2017](#); [Kahn et al., 2018](#)). Navigation implies planning a path, sometimes taking into account moving objects like humans or other robots, which has also been studied with DRL ([Everett et al., 2018, 2021](#)). This kind of approaches have been used for application with aerial vehicles ([Zhang et al., 2016](#)) as well as for autonomous driving ([Shalev-Shwartz et al., 2016](#); [Chen et al., 2020](#)). Approaches in these fields show promising

results, dealing with more and more complex environments and situations. But there are still issues with the great sample inefficiency of DRL models.

An interesting solution to this problem is sim2real approaches. As we have said in [Section 2.4](#), sim2real methods try to bridge the reality gap, finding techniques to adapt a model trained in simulation to the real world without losing its efficiency. This has been successfully done for static robots doing object grasping and object manipulation, mainly using domain randomisation ([Tobin et al., 2017](#); [Peng et al., 2018](#); [Andrychowicz et al., 2020](#)). In mobile robotics, some similar techniques have been used ([Tai et al., 2017](#); [Kang et al., 2019](#)) but there is still much to do to have reliable sim2real approaches.

Finally, DRL is studied for multi-robot applications. MADRL approaches with state-of-the-art DRL models can be used for navigation and collision avoidance ([Chen et al., 2017](#); [Long et al., 2018](#); [Semnani et al., 2020](#); [Xiao et al., 2020](#)) or information gathering ([Queralta et al., 2020](#)). These approaches are still under-developed and relatively simple in that they do not use state-of-the-art MADRL techniques. This shows that the field is still quite young and has room for future works. Complex methods that yield good results in MADRL can be impractical for robotic applications. So we need to find the best ones to deal with the issues of mobile robotics.

In conclusion, DRL is extensively studied in mobile robotics. However, research in DRL and MADRL is still very recent and has still a lot of potential for future works. The key is most likely in finding efficient sim2real solutions in order to solve issues of sample inefficiency and generalisation that often occur with DRL models.

3.1.4 Emergent communication

As described in [Section 2.1](#), communication can be a great tool in a multi-agent system to deal with partial observability. Research in this direction has started along with the development of the first MADRL approaches. A first kind of approach uses a centralised network for communication. This network is used to transmit messages between agents, which can be done in multiple different ways. In CommNet, [Sukhbaatar et al. \(2016\)](#) simply compute the mean of all messages and send the result to all agents. This approach has since been pushed further with more complex ways of transmitting the messages ([Peng et al., 2017](#); [Singh et al., 2019](#)). One particularly successful approach is TarMAC ([Das et al., 2019](#)), which uses attention to allow for targeted communication between agents. Other techniques look into decentralised execution ([Mordatch and Abbeel, 2018](#); [Jiang and Lu, 2018](#)) like DIAL which uses differentiable communication to learn to communicate directly through back-propagation of the task reward ([Foerster et al., 2016](#)). Finally, research in multi-agent communication has also been studying value factorisation, adapting VDN and QMIX to incorporate message sharing between agents ([Zhang et al., 2019](#); [Wang et al., 2020](#); [Zhang et al., 2020](#)).

All these approaches show interesting results. However, emergent communication using continuous signals still has major issues. First, measuring the effectiveness of the learnt language is quite difficult. [Lowe et al. \(2019\)](#) show that even if a language can provide information about the state of the environment, it will not necessarily impact the environment or the agents. Emergent languages lack crucial qualities of natural languages like distinct rules and compositionality, which makes them really difficult to interpret and arguably less efficient ([Kottur et al., 2017](#); [Lazaridou and Baroni, 2020](#)). In addition to that, learning and building a new language is a huge amount of complexity added to the already difficult process of MADRL. Thus, the benefit of sharing information may not be sufficient to justify the additional complexity. Therefore, in emergent communication, the communication system should be kept as simple as possible to ensure that it does not harm the learning process. However, the full potential of communication might only be achievable by learning a pre-existing, discrete communication system.

3.2 Creating a simulated environment

After reviewing past works and defining what issues we wanted to address, we started working towards our future contributions. The first step was to decide on an environment to set our experiments and a task to try solving. We quickly decided to focus on simulated environments, at least at first. However, as we said earlier, to be relevant the simulated task would have to reflect important issues of a real robotic task. It should feature partial observability and require full co-operation and coordination to be solved optimally. It should also be a task we could perform in the real world in case we want to address it in the sim2real part of the thesis. The environment should be continuous as any real-world environment is.

We settled on the task presented in [Figure 1](#). It features a team of agents that have to push a large object on the corresponding landmark to end the episode. The object is heavier than the agents, so they should coordinate to push it together in order to complete the task faster. Agents have a limited field of vision: they only see entities that are close to them so they have to explore the environment to know where the object and the landmark are. At each episode, the position of all entities are initialised randomly. Agents’ observations are composed of the position and velocity of themselves and other entities in their field of vision. Actions are either continuous or discrete. When continuous, agents can move any direction they want by producing a two-dimensional velocity vector. When discrete, agents can choose between five actions: move right, left, up or down, or do nothing. The reward at each time-step was initially defined as the opposite of the squared distance between the object and the landmark, plus a penalty of -10 for any collision between agents. As agents all receive this negative reward at each time-step, they should be incited to push the object on the landmark as quick as possible in order to maximise their total return. However, as we will see in [Section 3.3](#), this reward has not been sufficient for the baseline methods we trained. Therefore, as a sanity check, we designed a more aggressive reward that adds to the previous one the opposite of the average distance between agents and the object. This reward should motivate the agents to move toward the object quickly, and thus find that the object should be pushed toward the landmark. The exact definition of observations, actions and rewards are detailed in [Annex A](#).

The environment chosen is the Multiagent Particle Environment (MPE)¹. It was originally created by [Mordatch and Abbeel \(2018\)](#) for studying emergent communication. Since then, it has been used in many papers to create simple scenarios involving multiple agents ([Lowe et al., 2017](#); [Jiang and Lu, 2018](#); [Wang et al., 2020](#)). It is very convenient for training reinforcement learning models for multiple reasons. First, it is a really simple two-dimensional environment that provides us with an accessible version of the MADRL problem. Its code is relatively simple, so it is fairly easy to understand how it works and to modify it if needed. Second, it is extremely straightforward to use as it works like a OpenAI Gym environment², a framework for reinforcement learning environments that is widely used as it makes executing episodes really simple. In addition to that, its design fits our needs well: the environment is continuous and physics-based, which are important features of robotic environments. Lastly, MPE has already a set of basic scenarios available and ready to use, like the common predator-prey scenario widely studied in research. However, we found that these scenarios were too simple, lacking important components like partial observability and long-term planning. Therefore, we decided to build our own custom scenario, which is also easy to do with MPE. In conclusion, this environment is very convenient for our work. It is very basic but allows us to create an interesting scenario.

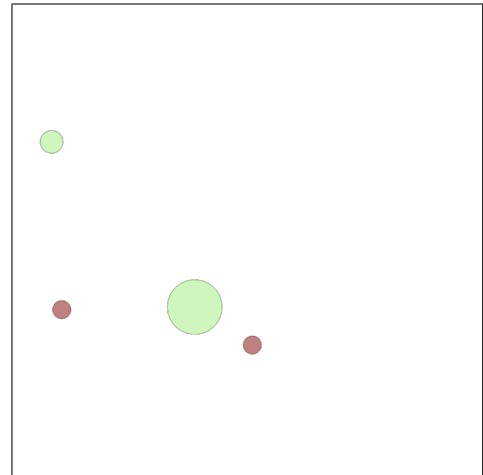


Figure 1: Cooperative push scenario. Agents (red circles) need to push the object (large green circle) on the landmark (small green circle).

3.3 Training methods from the literature

To start experimenting with our scenario, we chose three baselines to compare our future developments. The first one is MADDPG ([Lowe et al., 2017](#)), which is relatively easy to use and is still one of the best policy-based MADRL methods. The second is QMIX ([Rashid et al., 2018](#)), which is arguably the best state-of-the-art approach and will allow us to evaluate a value-based method with value factorisation. Finally, we also tried the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) to include a derivative-free direct policy search algorithm in our baselines. Making all those models work in code, along with developing a functioning scenario, has been tough and time consuming. But we now have a good base for future developments.

The first conclusion we found was that these baselines struggle in our scenario. There appears to be an exploration issue, where agents struggle to find that they need to push the object

¹<https://github.com/openai/multiagent-particle-envs>

²<https://gym.openai.com/>

through random actions. This motivated the creation of a simpler version of the environment with no partial observability, as a sanity check. This also led to the more aggressive reward that directly motivates the agents to go towards the object, inevitably pushing the object and finding out their ultimate goal. This reward gave us better results. MADDPG and QMIX have found near optimal strategies in the fully observable scenario. CMA-ES struggles more, probably because it needs more samples for training than the other two methods. However, this aggressive reward is inherently flawed as it makes the agents go towards the object when they sometimes have to explore the environment. We will have to find ways to work without this imperfect reward. The problem is certainly with the exploration strategy. Solutions might be to use more devised exploration strategy like MAVEN (Mahajan et al., 2019), design a reward shaping strategy or use approaches from the meta-learning field to make the agents understand their goal more explicitly.

4 Future steps

Now that I have built good foundations for this thesis, I need to start working towards future contributions. The very first step is to find a working reward function for the cooperative push scenario. The aggressive reward used before was a good way to test that the environment and our algorithms work. But it is essentially a trick to have good results with existing methods, which is not what we ultimately want. Moreover, it will certainly fail to work in the partially observable version of the environment, as agents will have to actively explore the environment. Thus, a simpler reward linked only to the object and the landmark needs to be designed. One option could be to add a large positive reward if the agents succeed in putting the object above the landmark.

Once we have a functioning reward, the next task will be to train agents with emergent communication. To show that communication is a crucial tool in multi-agent environments, its effect on performance must be demonstrated. We can draw inspiration from early works in emergent communication (Mordatch and Abbeel, 2018) to develop a simple communication system. It should allow agents to communicate part of their observations and policy, and be as simple as possible to avoid harming the learning process. We will experiment with this system in the partially observable version of the scenario and show how it helps overcoming the lack of information on the full state of the environment.

However, as we have said earlier, the real potential of communication might lie in the use of a pre-defined language, as described in Section 2.2. Therefore, the real contribution will be to train agents to use a language while learning our task. This will require a lot of development, first with the creation of a language agents can learn, and humans can interpret. Then, we will design a process to teach this language to agents. A combination of supervision and self-play should be efficient for agents to learn to use the language in the intended way, while exploring how it can be used in different situations. The goal will be to show that this language improves the agents' performance, and helps them generalise their knowledge to new situations.

Along with developing these contributions, we need to start publishing in conferences. As our subject is very broad, the field of conferences we can target is broad as well. We can aim for conferences in robotics, such as the International Conference on Robotics and Automation (ICRA). We can also target publications in artificial intelligence like the International Conference on Learning Representation (ICLR) and the conference on Neural Information Processing Systems (NeurIPS). Finally, we also have the conferences dedicated to MAS like the conference on Autonomous Agents and Multi-Agent Systems (AAMAS). All these examples are top-conferences, but we will also target smaller ones that might be easier to publish in.

5 Working in ECE's research lab

The entirety of my work during this thesis will be done inside the research lab of the engineering school ECE Paris. ECE funds this thesis entirely, providing me with a desk and any computer equipment needed. The majority of my time is dedicated to research. I meet every two weeks with my supervisor Jae-Yun Jun to present my work and discuss findings and future tasks. Every two months, we also meet with my two official thesis directors Nicolas Bredeche and Faïz Benamar, from *Institut des Systèmes Intelligents et de Robotique* (ISIR), to assess the current state of my work. All the presentations I have showed them, along with my bibliography and my codes,

are available on my Github repository³. Aside from my research work, I also have a teaching duty as part of my contract with ECE. I have to do a hundred hours of teaching per year with ECE students. Therefore, I have been teaching programming to first year students since September 2021.

Part of my work during this thesis is being a part of ECE's research lab. The research lab works in partnership with other laboratories in France and also organises the research course for ECE students in their 4th year. This has led me to interact with students and other researchers, first by presenting my work. I did multiple presentations to other researchers or to students, summarising the domains I will study in my thesis. I also had the opportunity of making a poster for the careers fair organised by ECE. The poster, showed in [Annex B](#), presents the context of my thesis and the four research directions presented in [Section 2](#). It was well appreciated by researchers and students, and helped me start conversations with students that were getting interested by research for their future career. This poster and the presentations I have done have been a good exercise for summarising and showing my work to people, which is a good training for future conferences I will participate in.

The research course introduces research to 4th year ECE students. They are invited to visit laboratories and they work on a research project throughout the year. I have taken part in this course as a mentor in a student project. I proposed a subject dealing with reinforcement learning and meta-learning. The goal is to try to apply some ideas of developmental psychology to artificial agents trained with reinforcement learning, trying to improve the sample efficiency of the learning algorithm. This subject is indirectly linked with my thesis. It allows me to explore other research ideas through the work of my group of students. For these students, they get to know how a research project is carried out and are introduced to interesting research fields.

6 Training plan

As part of my thesis, the doctoral school requires that I follow both scientific and nontechnical courses to complete my training. Due to my teaching work at ECE, I only need to do 50 hours of training during my thesis. For the scientific course, I chose to follow a course on mobile robotics given at *Sorbonne Université* during the second semester of 2021. Mobile robotics is the domain I was less familiar with starting this thesis. Thus, this course helped me understand how a mobile robotics problem is usually handled. For the non-technical courses, I chose courses to help me organise and present my work. I have already followed a course on scientific writing to train my written English skills and learn about how to write a paper. In the future, I will follow a course on organising my thesis and another on presenting my work orally in English.

References

- A. Akakzia, C. Colas, P-Y. Oudeyer, M. Chetouani, and O. Sigaud. Grounding Language to Autonomously-Acquired Skills via Goal Generation. In *ICLR 2021 - Ninth International Conference on Learning Representation*, 2021. URL <https://hal.inria.fr/hal-03121146>.
- C. Amato, G. Konidaris, L. P. Kaelbling, and J. P. How. Modeling and planning with macro-actions in decentralized pomdps. *Journal of Artificial Intelligence Research*, 2019.
- J. Andreas, D. Klein, and S. Levine. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the 34th International Conference on Machine Learning*, 2017. URL <http://proceedings.mlr.press/v70/andreas17a.html>.
- J. Andreas, D. Klein, and S. Levine. Learning with latent language. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018. doi: 10.18653/v1/n18-1197.
- M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 2020. URL <https://doi.org/10.1177/0278364919887447>.

³<https://github.com/Maxtoq/PhD-Thesis-Multi-agent-deep-reinforcement-learning-in-mobile-robotics>

- C. Berner, G. Brockman, B. Chan, V. Cheung, P. D biak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. J zefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang. Dota 2 with large scale deep reinforcement learning. 2019.
- Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.
- W. B hmer, V. Kurin, and S. Whiteson. Deep coordination graphs. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. URL <http://proceedings.mlr.press/v119/boehmer20a.html>.
- Y.-h. Chang, T. Ho, and L. Kaelbling. All learning is local: Multi-agent learning in global reward games. In *Advances in Neural Information Processing Systems*, 2004. URL <https://proceedings.neurips.cc/paper/2003/file/c8067ad1937f728f51288b3eb986afaa-Paper.pdf>.
- Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, 2019. doi: 10.1109/icra.2019.8793789.
- D. Chen, B. Zhou, V. Koltun, and P. Kr henb hl. Learning by cheating. In *Proceedings of the Conference on Robot Learning*, 2020. URL <http://proceedings.mlr.press/v100/chen20a.html>.
- Y. F. Chen, M. Liu, M. Everett, and J. P. How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation*, 2017. doi: 10.1109/ICRA.2017.7989037.
- C. Colas, T. Karch, N. Lair, J.-M. Dussoux, C. Moulin-Frier, P. Dominey, and P.-Y. Oudeyer. Language as a cognitive tool to imagine goals in curiosity driven exploration. In *Advances in Neural Information Processing Systems*, 2020. URL <https://proceedings.neurips.cc/paper/2020/file/274e6fcf4a583de4a81c6376f17673e7-Paper.pdf>.
- A. Das, S. Kottur, J. M. F. Moura, S. Lee, and D. Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.
- A. Das, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Neural modular control for embodied question answering. In *Proceedings of The 2nd Conference on Robot Learning*, 2018. URL <http://proceedings.mlr.press/v87/das18a.html>.
- A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau. TarMAC: Targeted multi-agent communication. In *Proceedings of the 36th International Conference on Machine Learning*, 2019. URL <http://proceedings.mlr.press/v97/das19a.html>.
- A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. Go-explore: a new approach for hard-exploration problems. 2019. URL <http://arxiv.org/abs/1901.10995>.
- A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. First return then explore. *Nature* 590, 2021. doi: <https://doi.org/10.1038/s41586-020-03157-9>.
- M. Everett, Y. F. Chen, and J. P. How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018. doi: 10.1109/IROS.2018.8593871.
- M. Everett, Y. F. Chen, and J. P. How. Collision avoidance in pedestrian-rich environments with deep reinforcement learning. *IEEE Access*, 2021. doi: 10.1109/ACCESS.2021.3050338.
- J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. URL <https://ojs.aaai.org/index.php/AAAI/article/view/11794>.
- J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016.

- M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg. Noisy networks for exploration. 2017.
- S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. 2018.
- D. Ha and J. Schmidhuber. World models. 2018.
- T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. 2018.
- D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. 2018.
- D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. 2019.
- M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. 2017.
- T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. Agapiou, J. Leibo, and A. Gruslys. Deep q-learning from demonstrations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. URL <https://ojs.aaai.org/index.php/AAAI/article/view/11757>.
- J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 2021. doi: 10.1177/0278364920987859. URL <https://doi.org/10.1177/0278364920987859>.
- S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- J. Jiang and Z. Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, 2018. URL <https://proceedings.neurips.cc/paper/2018/file/6a8018b3a00b69c008601b8becae392b-Paper.pdf>.
- G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018. doi: 10.1109/ICRA.2018.8460655.
- K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In *2019 International Conference on Robotics and Automation (ICRA)*, 2019. doi: 10.1109/ICRA.2019.8793735.
- S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.
- T. Karch, L. Teodorescu, K. Hofmann, C. Moulin-Frier, and P.-Y. Oudeyer. Grounding spatio-temporal language with transformers. 2021.
- S. Kottur, J. Moura, S. Lee, and D. Batra. Natural language does not emerge ‘naturally’ in multi-agent dialog. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.
- L. Kraemer and B. Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 2016. doi: <https://doi.org/10.1016/j.neucom.2016.01.031>.
- A. Lazaridou and M. Baroni. Emergent multi-agent communication in the deep learning era. 2020.
- A. Lazaridou, A. Peysakhovich, and M. Baroni. Multi-agent cooperation and the emergence of (natural) language. 2016.

- T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. 2015.
- P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation*, 2018. doi: 10.1109/ICRA.2018.8461113.
- R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017. URL <https://dl.acm.org/doi/abs/10.5555/3295222.3295385>.
- R. Lowe, J. Foerster, Y.-L. Boureau, J. Pineau, and Y. Dauphin. On the pitfalls of measuring emergent communication. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019.
- R. Lowe, A. Gupta, J. Foerster, D. Kiela, and J. Pineau. On the interaction between supervision and self-play in emergent communication. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=rJxGLlBtWH>.
- A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson. Maven: Multi-agent variational exploration. *Advances in Neural Information Processing Systems*, 32, 2019.
- V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. 2013.
- V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature* 518, 2015. doi: 10.1038/nature14236.
- I. Mordatch and P. Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. URL <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17007>.
- D. T. Nguyen, A. Kumar, and H. C. Lau. Credit assignment for collective multi-agent rl with global rewards. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pages 8113–8124, 2018. URL <http://papers.nips.cc/paper/8033-credit-assignment-for-collective-multiagent-rl-with-global-rewards>.
- K. Nguyen, D. Misra, R. Schapire, M. Dudík, and P. Shafto. Interactive learning from activity description. 2021.
- D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. 2017.
- P. Peng, Y. Wen, Y. Yang, Q. Yuan, Z. Tang, H. Long, and J. Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. 2017.
- X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018. doi: 10.1109/icra.2018.8460528.
- J. P. Queralta, J. Taipalmaa, B. C. Pullinen, V. K. Sarker, T. N. Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund. Collaborative multi-robot systems for search and rescue: Coordination and perception. 2020.
- T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. URL <http://proceedings.mlr.press/v80/rashid18a.html>.
- T. Rashid, G. Farquhar, B. Peng, and S. Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multiagent reinforcement learning. 2020.

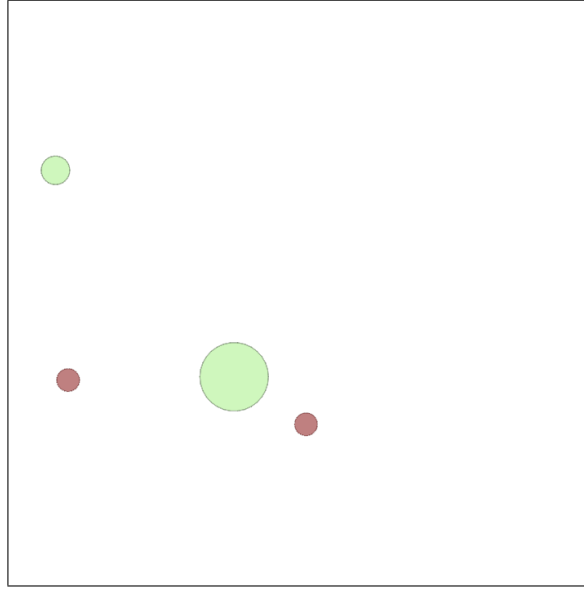
- T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *4th International Conference on Learning Representations*, 2016. URL <http://arxiv.org/abs/1511.05952>.
- J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering atari, go, chess and shogi by planning with a learned model. 2019.
- C. Schroeder de Witt, T. Gupta, D. Makoviichuk, V. Makovychuk, P. H. S. Torr, M. Sun, and S. Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? 2020a.
- C. Schroeder de Witt, B. Peng, P.-A. Kamienny, P. Torr, W. Böhmer, and S. Whiteson. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. 2020b.
- J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. 2015.
- J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. 2017.
- R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to explore via self-supervised world models. *Proceedings of the 37th International Conference on Machine Learning*, 2020.
- S. H. Semnani, H. Liu, M. Everett, A. de Ruiter, and J. P. How. Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning. *IEEE Robotics and Automation Letters*, 2020. doi: 10.1109/LRA.2020.2974695.
- S. Shalev-Shwartz, S. Shammah, and A. Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. 2016.
- L. S. Shapley. A value for n-person games. *Contributions to the Theory of Games*, 1953. URL <https://ci.nii.ac.jp/naid/10013542751/en/>.
- P. Shyam, W. Jaśkowski, and F. Gomez. Model-based active exploration. *Proceedings of the 36th International Conference on Machine Learning*, 2019.
- D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature* 529, 2016. doi: 10.1038/nature16961.
- D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. P. Lillicrap, K. Simonyan, and D. Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. 2017a. URL <http://arxiv.org/abs/1712.01815>.
- D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. P. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature* 550, 2017b. doi: 10.1038/nature24270.
- A. Singh, T. Jain, and S. Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. 2019.
- K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2019.
- S. Sukhbaatar, A. Szlam, and R. Fergus. Learning multiagent communication with backpropagation. *Advances in Neural Information Processing Systems*, 2016.
- P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018.

- R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999. doi: [https://doi.org/10.1016/S0004-3702\(99\)00052-1](https://doi.org/10.1016/S0004-3702(99)00052-1). URL <https://www.sciencedirect.com/science/article/pii/S0004370299000521>.
- N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, and P. Corke. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 2018. URL <https://doi.org/10.1177/0278364918770733>.
- L. Tai, G. Paolo, and M. Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017. doi: 10.1109/IROS.2017.8202134.
- A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente. Multi-agent cooperation and competition with deep reinforcement learning. *PLOS ONE*, 2017. URL <https://doi.org/10.1371/journal.pone.0172395>.
- M. Tan. Multi-agent reinforcement learning: Independent versus cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, 1993.
- J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017. doi: 10.1109/IROS.2017.8202133.
- H. van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016. URL <http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12389>.
- L. S. Vygotsky. *Thought and Language*. 1934.
- J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang. Qplex: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=Rcmk0xxIQV>.
- T. Wang, J. Wang, C. Zheng, and C. Zhang. Learning nearly decomposable value functions via communication minimization. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=HJx-3grYDB>.
- Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016. URL <http://proceedings.mlr.press/v48/wangf16.html>.
- E. Wei and S. Luke. Lenient learning in independent-learner stochastic cooperative games. *The Journal of Machine Learning Research*, 2016.
- D. H. Wolpert and K. Tumer. *Optimal Payoff Functions for Members of Collectives*, pages 355–369. 2002. doi: 10.1142/9789812777263_0020. URL https://www.worldscientific.com/doi/abs/10.1142/9789812777263_0020.
- Y. Xiao, J. Hoffman, T. Xia, and C. Amato. Learning multi-robot decentralized macro-action-based policies via a centralized q-net. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020. doi: 10.1109/ICRA40945.2020.9196684.
- S. Q. Zhang, Q. Zhang, and J. Lin. Efficient communication in multi-agent reinforcement learning via variance based control. In *Advances in Neural Information Processing Systems*, 2019. URL <https://proceedings.neurips.cc/paper/2019/file/14cfdb59b5bda1fc245aadae15b1984a-Paper.pdf>.
- S. Q. Zhang, Q. Zhang, and J. Lin. Succinct and robust multi-agent communication with temporal message control. In *Advances in Neural Information Processing Systems*, 2020. URL <https://proceedings.neurips.cc/paper/2020/file/c82b013313066e0702d58dc70db033ca-Paper.pdf>.

- T. Zhang, G. Kahn, S. Levine, and P. Abbeel. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016. doi: 10.1109/ICRA.2016.7487175.
- M. Zhou, Z. Liu, P. Sui, Y. Li, and Y. Y. Chung. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. 2020.
- Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation*, 2017. doi: 10.1109/ICRA.2017.7989381.

Annex

A. Cooperative push scenario



The Cooperative Push Scenario features a team of agents that need to push a big object on a landmark. Entities (i.e., agents, object and landmark) are spawned in random location at the start of an episode and are then trapped in the area we see in the image above, defined as $D = [-1, 1]^2 \subset \mathbb{R}^2$. The episode lasts for 100 time steps maximum. If agents manage to put the center of the object above the landmark, the episode stops.

At each time steps, the agents each receive an individual observation from the environment, which is a vector of real numbers composed of:

- the agent's absolute position $(x, y) \in D$ and velocity $(dx, dy) \in D$,
- the absolute position and velocity of all other agents and the object,
- the absolute position of the landmark.

The full dimension of the observation is: $\dim(X) = 4 * n_{agents} + 4 + 2$. In the example with two agents depicted above, the dimension of the observation will be 14. In the partially observable version of the environment, entities that are too far from the agent are masked with zeros in the input.

Actions are either:

- Discrete: agents choose between five possible actions: go right, left, up, down or do nothing. The dimension of the model's output is 5.
- Continuous: agents choose the direction they want to go. The dimension of the output is 2.

All agents receive the same reward at each time step, except for the collision penalty that concerns only the agents that are colliding. For now, we are working with two definitions of the reward:

- the initial reward, which we call obj , is defined as the opposite squared distance between the object and the landmark, plus the penalty for collision ρ :


$$R_{obj}^t = -dist(object, landmark)^2 + \rho,$$

- the second reward called $distrew$ (for "distance reward") is the obj reward minus the average distance between the agents and the object:

$$R_{distrew}^t = R_{obj}^t - \frac{1}{n_{agent}} \sum_{k=1}^{n_{agent}} dist(agent_k, object).$$

The function $dist$ refers to the Euclidean distance. For previous experiments, the collision penalty ρ was fixed at -10.

B. Poster for ECE's careers fair




ECE
ÉCOLE D'INGÉNIEURS
ENGINEERING SCHOOL

Multi-agent deep reinforcement learning in mobile robotics

Maxime Toquebiau^{1,2}, Nicolas Bredeche², Faiz Benamar², Jae-Yun Jun¹

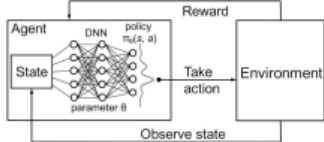
¹ECE Paris,
²Sorbonne Universités, CNRS, ISIR

Phd Thesis started in January 2021.



ISIR
INSTITUT
DES SYSTÈMES
INTELLIGENTS
ET DE ROBOTIQUE

Deep reinforcement learning (DRL)



Goal
Maximise long-term cumulative reward

Tools

- **Deep learning**: extract valuable information from observations
- **Reinforcement learning**: learn a policy (sequence of actions) that maximises the reward

Three research dimensions


Multi-agent systems (MAS)

Multiple agents interacting with one another and with their environments.

Issues

- **Complexity**: More agents means that the joint spaces of states and actions are bigger and harder to process.
- **Non-stationarity**: With multiple agents constantly learning new behaviours, the environment is endlessly evolving.
- **Partial Observability**: Each agent's reward depends on events it can't observe.
- **Credit assignment**: All agents in a team have a different impact on the reward received at a team-level. We need ways to divide the reward based on individual performance.

Mobile robotics



Constraints

- Complexity of real-world environments
- Imperfections of sensors and actuators
- Safety
- Limited communication
- Dynamic environments

Four research directions


Problematics

- How to design more relevant simulated environments ?
- How to use deep reinforcement learning in real-world environments ?
- How to use and interpret the actions of artificial agents ?
- How to interact with robots ?
- How to use a policy learnt in simulation on a real-world robot ?

Emergent communication

Context
Despite its clear potential, communication isn't used enough in the multi-agent DRL literature.

Contribution
Study emergent communication in MAS and their impact on performance.



Reference:
Mordatch and Abbeel et al., *Emergence of Grounded Compositional Language in Multi-Agent Populations*, AAAI, 2018.

Learning an existing language

Context

- Robots will need to **interact** with human beings, and thus communicate with an **interpretable language**.
- Using a pre-existing and discrete language can not only help communicate with humans, but also drive the understanding of the environment.

Contribution
Design a language that can be learnt by artificial agents by interaction and that improve their performance in a chosen task.

Reference:
Luketina et al., *A Survey of Reinforcement Learning Informed by Natural Language*, IJCAI, 2019.

Multi-agent credit assignment

Context

- In a cooperative multi-agent setting, the global reward isn't suitable for agents to learn their optimal policy, as they are not able to deduce the quality of their actions from the reward obtained at a team level.
- Communication takes part in finding the best solution to a given task, thus it must be taken into account when assigning credit to agents.

Contribution

- Study existing solutions and find the best suited for our context.
- Link credit assignment to communication.

Reference:
Foerster et al., *Counterfactual Multi-Agent Policy Gradient*, AAAI, 2018.

Macro-actions as a sim-to-real approach


Context

DRL methods are notoriously hard to apply in real world environments due to:

- very low sample efficiency,
- need to experience very bad outcomes to find the optimal policy.

Contribution
Use macro-actions to learn high-level policies in simulation and transfer them easily in the real world.

Reference:
Xiao et al., *Learning Multi-Robot Decentralized Macro-Action Based Policies via a Centralized Q-Nat*, ICRA, 2020.



Learn in simulation → Transfer to reality

Contact: maxime.toquebiau@ece.fr