Communication strategies in Swarm Robots Search and Rescue scenarios using Deep Reinforcement Learning

Giacometti Giovanni
giog@itu.dk

Barbiero Alessandro
aleba@itu.dk

Carausu Toader
toca@itu.dk

December 16, 2022

Abstract

Swarm Robotics is a field of robotics that aims at developing individual robots able to solve complex tasks when acting in a group. Researches in the area address the design of distributed controllers to achieve swarm behaviors. This paper explores the application of Deep Reinforcement Learning techniques to Swarm Robotics, focusing on defining stimulating environments and suitable communication patterns that can enhance the performance of the swarm in Search and Rescue settings.

1 Introduction

Swarm robotics investigates how a large number of robots can work together to achieve a common goal using decentralized control and possibly communication. It is understandable that a large number of simple robots can perform complex tasks in a more efficient way than a single robot. The behavior of this kind of robot has been widely studied in different tasks like aggregation, dispersion, pattern formation, coordinated motion, collective transport of objects, collaborative mapping, consensus achievement, task allocation, and source search [1, 2].

Despite the relative youth of the topic, there already exist real-world cases in which Swarm Robotics has been applied [3].

A scenario in which swarm robotics can make a difference is Search and Rescue (SaR). When timing is crucial and the conditions of the environment are dangerous, having a swarm of robots capable of finding a target can be really helpful. A real-world application of swarm technologies in this field is represented by the GUARDIANS project [4], where robots support firefighters during the search by keeping defined pattern formation thus helping them avoid obstacles in low visibility environments. Our environment mimics a simulated disaster, with a maze representing a destroyed building and a red cube representing a motionless victim (Figure 2). Our research will concentrate only on the Search part, the most time-consuming for human rescuers. We will explore how communication and swarm size can affect the performance of a group of decentralized agents in mazes of different dimensions, using Proximal Policy Optimization (PPO) learned policies to achieve the desired results.

2 Background

The literature contains numerous studies about swarm robots and the strategies employed to simulate their collective nature. Manually created algorithms, where the best controller is found by following heuristics and iteratively adjusting parameters, are currently the most used approach to define the behavior of agents. Such algorithms include Particle Swarm Optimization (PSO) [5] and Glowworm Swarm Optimization (GSO) [6]. These patterns proved to be successful in simple and static environments but struggles to adapt in complex and variable scenarios. This is due to the complexity of modeling interactions undergoing between robots while they're solving a structured task [7].

In contrast to this approach, our work leverages Deep Reinforcement Learning (DRL) techniques to make agents discover their controllers autonomously. The process through which learning is achieved in Reinforcement Learning settings is based on iterative interactions with the environment. The agent observes the environment and performs an action following a policy. The environment receives the action and emits a reward. The goal of the agent is to adjust its policy in order to maximize the cumulative reward in a defined time period. DRL exploits Deep Neural Networks to enhance the observation complexity of the environments the agent can handle.

There exist previous works that make use of DRL algorithms to achieve swarm behaviors in SaR setting [8, 9]. The difference between them lies in the perceiving ability of individual robots, the algorithms used to train policies, and swarm sizes. Na et al [10] proposed a Federated Learning training paradigm based on Deep Deterministic Policy Gradient, with the aim of tackling limited communication bandwidth issues. The robots they developed are aware of the position of the target from the beginning of the exploration. It differs from our implementation since the agents we developed are not conscious of the target location until they can physically see it, shifting the focus on its localization in a complex maze rather than coping with low-capacity communication. Tan et al [11] exploited Proximal Policy Optimization with Convolutional Neural Networks to build robots able to navigate and scan an environment. The number of agents acting at the same time was 3, while our research investigates performances of different swarm sizes, up to 20.

Multiagent communication is also a widely explored topic. Huttenrauch et al [12] developed swarm robot systems characterized by different local communication patterns, including a histogram of distances to observed neighbored agents. The resulting vector incorporates information related to different agents while maintaining fixed dimensionality. Foerster et al [13] explored the learning of communication protocols in complex environments where communication is required to solve tasks. As a downside, the communication is limited to a set of discrete values that agents have to select upon, while all our methods implement real-valued messages.

2.1 PPO

Policy gradient algorithms are a class of RL techniques where the policy is updated through gradient descent methods to maximize the expected reward of the agent. PPO is a model-free policy gradient algorithm widely employed in RL tasks, including cooperative multiagent games [14]. It builds upon the ideas of Trust Region Policy Optimization algorithm but it constrains policy updates by directly clipping probability ratios in the objective function. It achieves a balance between implementation, sample complexity and ease of tuning [15].

3 Game mechanics

3.1 Assumptions

In order to create our environment we took into account the following assumptions:

- The agent is assumed to be capable of recognizing the target for which it is searching. Perceiving a victim from sensory data is still a difficult task for robots, but that is beyond the scope of this research.
- The environment is unknown to the agent with an unknown probability that each path will lead to the target location.
- The agent is constrained by time. A search robot has to find a victim in time or within the battery limits.
- Agents have only local communication and sensing capabilities. The coordination is therefore distributed.

3.2 The game

The purpose of the game is straightforward: reach the target inside the maze in the shortest possible time.

Reinforcement Learning can be a crucial element in this scenario since optimal strategies are complex to design a priori whereas AI-developed policies are better suited to generalize to different mazes.

3.2.1 Controls

The agents can be controlled with different types of actions:

- (Continuous) North/South - West/East
- (Continuous) Rotation - Translation
- (Discrete) Seven actions [stop - forward - backward - left - right - turn left - turn right]

3.2.2 Training Scene

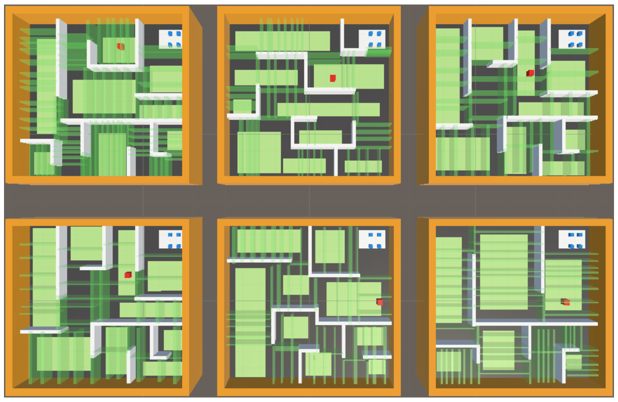

Figure 1: Training Environment: 6 mazes pseudorandomly changing walls and target positions with a swarm of 4 agents.

The training scene has a dimension of 100x100 and works in the following way:

- A swarm of 4 agents is spawned in a Random maze selected among 5 different mazes
- The maze changes every time the agents reach the target
- The position of the target is randomly changed every time the agents get to it, and it can spawn all over the maze
- The starting position of the agents is randomly selected inside a predefined area in the maze (The white one in Figure 1)
- Checkpoints are removed once they are traversed by an agent
- The episode is terminated after 10000 steps if the agents can't reach the target

3.2.3 Test Scene

We built 4 different mazes of increasing complexity to test our models. Figure 2 shows the widest test maze. Each test maze contains a number of targets that is proportional to its size.

The simulation we implemented deploys the model on different swarm and handles the testing in all mazes. The procedure is as follows: it starts from the smaller maze and the smaller swarm. A new target is spawned when an agent reaches the target or when the episode times out. Once the last swarm is tested, the simulation proceeds to the next maze, bigger than the previous one. The test ends after the most numerous swarm is tested in the largest maze.

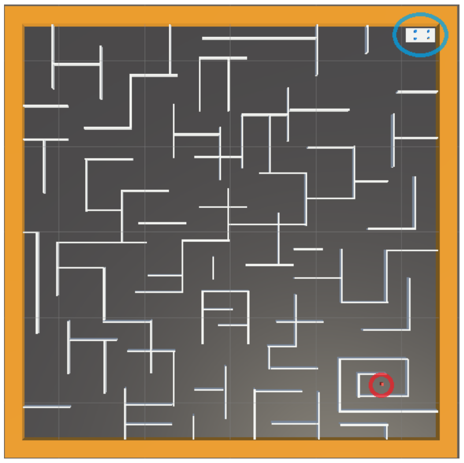

Figure 2: Big maze (500x500) used to test the behavior of the swarms. The blue circle points at the swarm position, the red one at the target.

4 Methods

4.1 PPO settings

ML-Agents Unity Package provides an implementation of the PPO algorithm. It runs in a separate Python process communicating with the running Unity application over a socket.

PPO parameters can be set using YAML configuration files. Its hyperparameters are set as follows and apply to all models: learning rate is 0.0003, entropy constant beta is 0.0005, clipping parameter epsilon is set to 0.1, lambda to 0.99. Actor and critic neural network architectures are composed of 2 hidden layers of 256 neurons each.

4.2 Environment Design

The observation space of each agent is formed by a raycast-based view and a communication list. The view consists of 15 rays of length 25 meters shot from the front of the agent with a field of view of 140 degrees. Each of them can recognize the object it collides with - *Agent, Wall, or Target* - and the distance to that object. The communication list is a vector whose size depends on the communication type and contains values exchanged by the agents. Communication methods are discussed in section 4.3. The actual input agents receive is ultimately obtained by stacking the current observation with the previous one. The choice of stacking two observations is due to the tradeoff between the willingness of conveying the idea of movement and keeping the input dimensionality in a reasonable range.

As stated in 3.2.1, the simulation we built offers the possibility to choose among three types of action. Eventually, we considered agents moving through discrete actions.

RL agents learn policies by interacting with the environment and leveraging on the rewards it produces. Tuning rewards represents an important degree of freedom that we tweaked to induce the intended behavior. Rewards obtained by agents at each time step are given by the sum of the following rewards:

- Checkpoint crossing. Its purpose is to enhance the exploration of the maze.
- Reach the target. It represents the goal of the simulation. Once an agent gets to the target, this reward is assigned to all the agents of the swarm.
- Target Proximity. This reward is inversely proportional to the distance to the target and an agent receives it if the target is in sight. Its aim is to encourage the agent to move toward the target.
- Hit Wall. Negative reward assigned when an agent hits a wall.
- Hit Agent. Negative reward assigned to agents when they bump into each other.
- Existential. Negative reward assigned at each step to boost agents speed in solving the task.
- Agent Proximity. Negative reward inversely proportional to the distance to other agents if they can see each other. The goal is to inspire the agents to spread as much as possible.

4.3 Communication

Communication between individual robots plays a key role in solving cooperative tasks. We attempted to investigate how it actually impacts the performance of the swarm, also in relation to its size. We implemented Communication with n nearest neighbors, achieving what in [16] is referenced as *Comm-Near* communication. In our experiments, we set the value to 3. This aims at creating realistic models, since communication with the whole swarm is infeasible in real life, and accomplishes scalability since a model trained with at least 4 agents can be deployed in swarms of any dimension. Several types of communication were explored:

- **Distance Communication**: the communication list is populated with the distances of each agent from the others. The idea is to help agents in determining the direction where to move based on closeness with the others. List size is 3.
- **Positions Communication**: the communication list contains the position of each agent. Values are normalized to adapt to any environment. The idea is to foster the creation of a "map" of the part of the environment where the agent is. Each agent contributes with x and z positions since agents cannot move along the y-axis. List size is 8.
- **Free Communication**: this communication is slightly different from the others. The PPO parameters are modified in order to output a continuous value besides the action to perform. At each step, all the agents collect messages from neighbors and store them in the communication list. Each agent communicates the same message to all its neighbors. The goal of this communication is to explore whether agents can learn to send each other meaningful messages that actually influence their behavior. List size is 3.
- **Mixed Communication**: we combined different communication methods to see if performance would improve.

4.4 Training

Training occurs in the environment described in section 3.2.2. It is always conducted with 4 agents since models can be deployed in any kind of swarm. As figure 1 shows, it is performed in parallel in 6 areas to speed up the training time. Three are the sources of randomicity that enhance the generalization capability of the agents: the maze in which the swarm is trained changes every episode, the target is always located at a different point and the agents start their task in a random position inside the spawn area.

4.5 Testing

We designed 4 different test areas to evaluate our models. These mazes are different from the testing ones in order to see which communication strategy and training parameters are able to better generalize the behaviour and thus find a target in a different environment with respect to the one in which the model was trained. Table 1 shows the number of targets and the dimension of each test maze. Test mazes are of increasing complexity and the number of targets is proportional to the size of the maze.

Name	Dimension	Number of targets
Toy	50x50	3
Small	100x100	6
Medium	300x300	18
Big	500x500	30

Table 1: Test Mazes used to evaluate performances. The number of targets is proportional to the size of the maze.

The starting position of the agents and target locations are fixed in each maze in order to compare the performances of different trained models. Once the test gets executed, the model under examination is deployed and tested in 5 different swarms, having respectively 4, 8, 12, 16, and 20 agents. The time to reach the target is the main criterion the evaluation is based on. We also keep track of the exploration rate of the maze to compare percentages of exploration in case a model is not able to locate the target. The test counts as failed if the agents cannot reach the target within 300 seconds.

4.5.1 Our Parameters

After several experiments, we came up with a base configuration that we used to train and compare all our models. Rewards are set as follows:

- Checkpoint crossing: 0.15
- Reach the target: 1 (shared)
- Target Proximity: 0.002
- Hit Wall: -0.01
- Hit Agent: -0.003
- Existential: -0.001
- Agent Proximity: -0.01

5 Results

The first variable we explored is the effect of increasing the number of robots in the swarm. Data concerning swarms that are not employing any communication method are represented in the bar plot of figure 3.

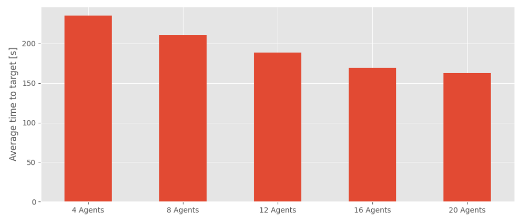

Figure 3: Average Time to reach the target in swarms of increasing size.

The y-axis represents the mean over all tests, the x-axis contains the swarm dimensions considered. The plot illustrates that increasing the number of robots enhances the exploration capabilities of the swarm, which is the expected behavior. However, the results are unsatisfactory. Mean values are significantly high and the percentage of unfound targets is not acceptable either, even for the swarm composed of 20 robots, as shown in table 2.

Swarm size	Percentage of unfound targets
4 Agents	77.19%
8 Agents	68.42%
12 Agents	59.64%
16 Agents	50.87%
20 Agents	49.12%

Table 2: Percentage of unfound target in swarms that are not communicating.

5.1 Communication

This section investigates the impact of communication methods on the performance of the robots. The distributions of time to target metric of each com-

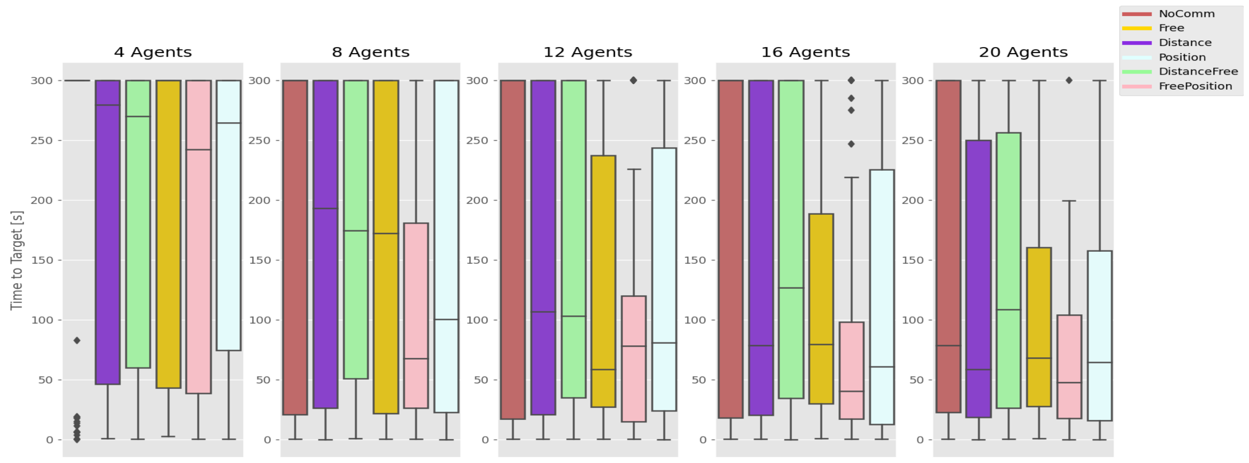

Figure 4: Boxplots showing distributions of time to target in swarms with different communication strategies.

munication method in different swarms are shown in figure 4. We considered the model without communication as the benchmark to evaluate the other behaviors.

Data support the hypothesis that communication helps swarms in solving their task. The contribution is particularly visible in swarms of higher dimensions.

Percentages of undetected targets with different communication methods support our claim as well. Values related to 20 agents swarms acting in the largest test maze are shown in figure 5.

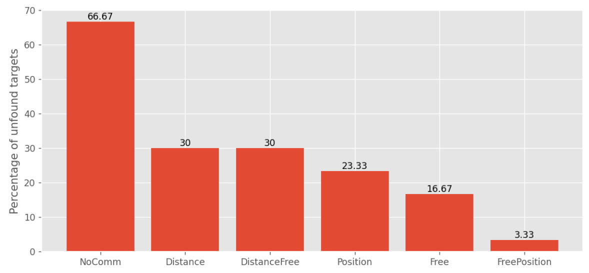

Figure 5: Percentage of Unfound targets by 20-agents swarms with different communication methods in the big maze tests.

Interestingly, the swarm implementing Free and Distance communication together is characterized by a percentage of unfound targets considerably lower than the one achieved by the swarm not communicating (30% against 66%). However, its median time-to-target appears worse than the no-communication one, as depicted in the last subplots of figure 4. The aim of our work is to create swarm robots acting in SaR settings, where speed has to be an essential feature of the search. Apparently, swarms that are communicating using both free messages and distances are more precise than swarms that are not communicating but sometimes they lack the necessary speed in pursuing their goal.

The other metric we computed in our tests is the exploration rate. It serves the purpose of evaluating models when neither of them is able to reach the target within the time limit. Figure 6 shows the percentage of map explored by each model in the only test where no swarm was able to reach the target. Data refer to swarms composed of 8 agents.

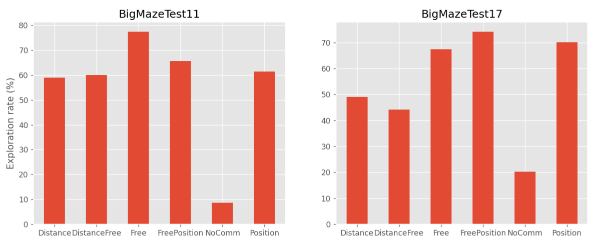

Figure 6: Exploration rates of models deployed in 8-agents swarm in tests where none of them was able to reach the target.

Overall, the results show that models implementing communication methods are more efficient than those without communication. Improvements are consistent and visible over all the metrics. Communication methods that yield the best results are free, position, and their combination. Communicating positions is reasonably functional to support agents behaviors since it conveys an idea of the local distribution of the swarm in the map. Moreover, it's the only technique that includes values directly related to agent movement, namely its position on x and z axis. Free communication contribution is more complex to perceive. The next section investigates more deeply its inputs in order to infer features of its exotic nature.

5.2 Free Communication

Free communication is based on the idea of devising a communication protocol that agents can exploit to aid each other. The complexity of this challenge is given by the need of the agents to agree on the meaning of messages. A useful message sent by an agent is deemed as positive only if the other agent perceives it in the correct way and achieves a positive group reward. If he fails to do so, the sender is not induced to send the message again [13]. Nevertheless, the free communication method is showing promising results in our experiments. We analyzed the signals exchanged by agents to understand if meaningful messages are actually communicated. We tested the model two more times, twisting in different ways the messages exchanged by the agents: first, we added a random noise between -0.5 and 0.5 to the generated message, then we substituted entirely the message received by the agent with a random value between -1 and 1, which is the range of the messages created by the agents. Data showed that distorted messages performed similarly to the base ones. The t-test run on the samples was not able to reject the null hypothesis and therefore we cannot say that the distributions obtained in the two test settings are different. The t-test carried out with the random values produced a *p-value* of *0.4*, while the test with the disturbed ones has a *p-value* of *0.9*. To analyze even further the usage of this message, we repeated the experiment forcing all the messages to zero. With this new scenario there is statistical significance to affirm that the performance is negatively impacted (*p-value 0.0002*), showing that communication is effectively exploited by the agents. Metrics distributions of all the mentioned trials are shown in figure 7.

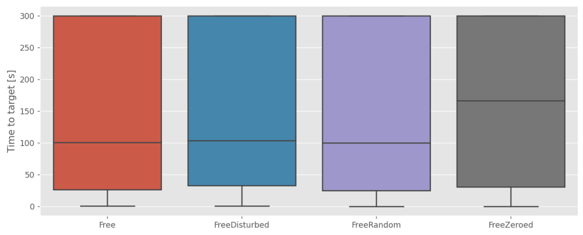

Figure 7: Comparison between Free communication and distorted versions performances.

As already stated, disrupting free communications doesn't impact the performance of the swarms. Still, agents rely on those messages, since setting them to zero exacerbates their ability. Our educated guess is that free communication acts as a source of randomness for the agent, which induces them to move and explore the maze, hence resulting in higher coverage of the map. Adding noise has been confirmed as valuable support during RL training both in Action-Space and Parameter-Space in order to enhance exploration [17]. Therefore, it is possible that by including a new source of randomness in the observation the trained agent exploration capability improves as well.

6 Discussion

Analyzing the results, it can be deduced that even with limited observation a numerous swarm of communicating robots is frequently capable of solving a difficult task in a partially observable environment. In addition, it is possible to affirm that RL agents can learn policies and leverage communication techniques to improve in non-fully cooperative tasks such as Search and Rescue. With this aim, the fine-tuning of shared and nonshared rewards is fundamental both for obtaining a good learning curve and for the final performance. Moreover, a relevant achievement is that with PPO algorithm it is feasible to learn complex policies in a small multiagent scenario and transfer that knowledge to more complex environments with even more agents.

However, the results are still not optimal. There is no model that is able to find all test targets and visual simulations show that the behavior of single robots is occasionally illogical. All our claims have to be considered in the context of a larger picture.

Several are the potential directions to investigate when seeking improvements in this task. Some of them have been explored and dropped in the early stages of the study for poor results or time complexity issues but surely deserve further attention. The main branches to explore are four.

6.1 Memory

Swarm Agents solutions can be made to store the data between consecutive steps or even episodes. There are numerous ways to create a personal or collective memory, as well as the information to be stored within. The main possible ways to implement a sort of memory in a Deep Neural Network agent that could be explored in the future are:

- Pheromones: This is the most appealing solution for a task like ours. It consists in the releasing of a steady substance that can make the other agents aware that somebody already explored a certain area. We decided to not use this kind of shared information to focus more on direct communication.
- LSTM / RNN: We tried this solution in the early stages but it did not yield any results.
- Stack more observations: It would mean increasing the observation space and therefore increasing the time necessary to train a valuable model.
- Custom memory structure

6.2 Algorithms and Continuous actions

During the development of our solution, we explored other algorithms for Agent Reinforcement Learning. The algorithms we considered are Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG). Both were eventually dropped because they lacked training speed, despite generating promising results.

DDPG was mainly used when we were initially considering the usage of continuous actions in order to better resemble real robot controls. Further studies can be also made on this path.

6.3 Different Communications

We mainly focused on the type of communication known as *interaction via communication*, [18] but literature contains honorable mentions regarding *interaction via sensing* category, such as pheromones. In terms of interaction via communication, other types of agent-learned communication could have been explored, such as a discrete 4 bits communication inspired by the one used by Kasai et al. [19].

6.4 Enhance randomicity in training

There exist different techniques that aim at maximizing the ability of the agent to explore. The most important are:

- Curiosity-driven exploration: The reward function is intrinsic to the agent that is obtaining rewards when exploring new states.
- Maximum Entropy: In this framework, the actor aims to maximize the expected reward while also maximizing entropy. It means trying to succeed at the task while acting as randomly as possible.

7 Conclusion

This paper investigates the application of Proximal Policy Optimization algorithm to Swarm robotics, with the purpose of building efficient and versatile policies capable of solving Search and Rescue tasks. We explored the impact of employing different swarm sizes and different types of communication. Our results prove that communication indeed helps robots in solving cooperative tasks. Position communication, especially when combined with free communication, emerges as the most promising one. In the scenario where the position data is not reliable, the swarm agents can still exploit Free Communication model for satisfactory results. Future works will concentrate on expanding their features and exploiting their potential.

References

[1] Manuele Brambilla et al. "Swarm robotics: a review from the swarm engineering perspective". In: *Swarm Intelligence* 7.1 (2013), pp. 1–41.

[2] Iñaki Navarro and Fernando Matía. "An introduction to swarm robotics". In: *International Scholarly Research Notices* 2013 (2013).

[3] Micha Sende Melanie Schranz1 Martina Umlauft and Wilfried Elmenreich. *Swarm Robotic Behaviors and Current Applications.* Tech. rep. 2020. URL: https://www.frontiersin.org/articles/10.3389/frobt.2020.00036/full#:~:text=It%20can%20be%20used%20to,or%20establish%20a%20communication%20network.&text=Coordinated%20motion%20moves%20the%20swarm,be%20arbitrary%20as%20in%20flocking..

[4] Joan Saez-Pons et al. "Multi-robot team formation control in the GUARDIANS project". In: *Industrial Robot: An International Journal* 37.4 (2010), pp. 372–383.

[5] Russell Eberhart and James Kennedy. "A new optimizer using particle swarm theory". In: *MHS'95. Proceedings of the sixth international symposium on micro machine and human science.* Ieee. 1995, pp. 39–43.

[6] Micael S Couceiro et al. "Benchmark of swarm robotics distributed techniques in a search task". In: *Robotics and Autonomous Systems* 62.2 (2014), pp. 200–213.

[7] Mauro Birattari et al. "Automatic off-line design of robot swarms: a manifesto". In: *Frontiers in Robotics and AI* 6 (2019), p. 59.

[8] Maximilian Hüttenrauch, Sosic Adrian, Gerhard Neumann, et al. "Deep reinforcement learning for swarm systems". In: *Journal of Machine Learning Research* 20.54 (2019), pp. 1–31.

[9] Toshiyuki Yasuda and Kazuhiro Ohkura. "Collective behavior acquisition of real robotic swarms using deep reinforcement learning". In: *2018 second IEEE international conference on robotic computing (IRC).* IEEE. 2018, pp. 179–180.

[10] Seongin Na et al. "Federated Reinforcement Learning for Collective Navigation of Robotic Swarms". In: *CoRR* abs/2202.01141 (2022). arXiv: 2202.01141. URL: https://arxiv.org/abs/2202.01141.

[11] Ziya Tan and Mehmet Karaköse. "Proximal Policy Based Deep Reinforcement Learning Approach for Swarm Robots". In: *2021 Zooming Innovation in Consumer Technologies Conference (ZINC).* 2021, pp. 166–170. DOI: 10.1109/ZINC52049.2021.9499288.

[12] Maximilian Hüttenrauch, Adrian Šošić, and Gerhard Neumann. "Local communication protocols for learning complex swarm behaviors with deep reinforcement learning". In: *International Conference on Swarm Intelligence.* Springer. 2018, pp. 71–83.

[13] Jakob Foerster et al. "Learning to communicate with deep multi-agent reinforcement learning". In: *Advances in neural information processing systems* 29 (2016).

[14] Chao Yu et al. "The surprising effectiveness of ppo in cooperative, multi-agent games". In: *arXiv preprint arXiv:2103.01955* (2021).

[15] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[16] Gregory Dudek et al. "A taxonomy for swarm robots". In: *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93).* Vol. 1. IEEE. 1993, pp. 441–447.

[17] Matthias Plappert et al. "Parameter space noise for exploration". In: *arXiv preprint arXiv:1706.01905* (2017).

[18] Levent Bayindir and Erol Şahin. "A review of studies in swarm robotics". In: *Turkish Journal of Electrical Engineering and Computer Sciences* 15.2 (2007), pp. 115–147.

[19] Tatsuya Kasai, Hiroshi Tenmoto, and Akimoto Kamiya. "Learning of communication codes in multi-agent reinforcement learning problem". In: *2008 IEEE conference on soft computing in industrial applications*. IEEE. 2008, pp. 1–6.