

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Introduction . . . . .	4
<b>2</b>	<b>Swarm Robotics</b>	<b>5</b>
2.1	Swarm Intelligence and Social animal inspiration . . . . .	6
2.2	Swarm robotics and multi robot systems . . . . .	6
2.3	Swarm robotics . . . . .	6
2.4	Swarm robotics properties . . . . .	7
2.5	Domain of application . . . . .	7
2.5.1	Tasks that cover a region . . . . .	8
2.5.2	Search and rescue missions . . . . .	8
2.5.3	Cleaning of oil speals . . . . .	8
2.5.4	Exploration . . . . .	8
2.5.5	Agriculture . . . . .	8
2.6	Swarm robotics basic tasks and problems . . . . .	8
2.6.1	Aggregation . . . . .	9
2.6.2	Dispersion . . . . .	9
2.6.3	Pattern formation . . . . .	9
2.6.4	Cordinated movement . . . . .	9
2.6.5	Hole avoidance . . . . .	9
2.6.6	Foraging . . . . .	9
<b>3</b>	<b>Deep reinforcement learning</b>	<b>10</b>
3.1	Introduction . . . . .	11
3.2	Machine learning . . . . .	11
3.3	Reinforcement learning . . . . .	11
3.4	Q learning . . . . .	12
3.5	Deep Q learning . . . . .	13
<b>4</b>	<b>Problem formulation And Solution Conception</b>	<b>14</b>
4.1	Introduction . . . . .	15
4.2	Goal defintion . . . . .	15
4.3	Problem Formulation . . . . .	15
4.3.1	Environement . . . . .	15
4.3.2	Shape generation . . . . .	16
4.4	State Representation And Reward Design . . . . .	16
4.4.1	Introduction . . . . .	16

4.4.2	State Representation . . . . .	16
4.5	Roboting Operating system . . . . .	16
4.5.1	Ros Basics . . . . .	16
4.6	State Of the art . . . . .	18
4.7	DQN . . . . .	18
4.8	DDPG . . . . .	18
4.9	Thier usages in multi agent pros and cons . . . . .	18
4.10	problem formulation and solution . . . . .	18
4.11	we opt for maddpg decentrilzed excution . . . . .	18
4.12	we opt for maddpg for better perfomance and scalability . . . . .	18
4.13	ROS and gazebo . . . . .	18
4.14	Conception of the solution . . . . .	18
4.15	Rewards desgin . . . . .	18
4.16	Networks Desgin . . . . .	18
4.17	talk about how you choose the hyperparameters . . . . .	18
4.18	how you calculeted tha angle and how lds works . . . . .	18
4.19	implemtation of the solution . . . . .	18

# **1 Introduction**

## 1.1 Introduction

These days, mobile robots have taken place in many fields like industry automation, planetary exploration, entertainment, and construction ..., for their ability to work in extreme environments with high precision and without fatigue[1]. Even so, a robot occasionally needs the support of other robots because it is impossible or difficult for them to perform some tasks on their own. For that, a new field has emerged to deal with these problems, swarm robotics.

Swarm robotics is relatively a new research topic that has gained more attraction in the last few years. It is about studying how a large number of simple robots (a swarm) can collaborate and work together to achieve predefined objectives and tasks that are often difficult or impossible to do for a single robot.[2].

One of the main challenges that swarm robotics researchs face, is pattern formation. Where the agents (robots) try to form different geometric shapes like squares, triangles and circles in order to perform a specific task.

We can solve this problem using two different approaches. The first one is a Centralized method where there exist a central unit which controls the swarm and give global state access. However, implementing this approach can be costly and less robust to failures. The second approach is a decentralized one, where each robot have uses local communication and have access only to his local state.[3]

Our primary objective in this thesis is to implement an RL algorithm in a system made up of a group of robots in order to form some specific geometric patterns using a decentralized method. Each robot will only have access to its local state and will interact and communicate with its neighbors in order to form the desired shapes.

## 2 Swarm Robotics

## 2.1 Swarm Intelligence and Social animal inspiration

Social animal and insects behavior in groups like the bees dancing, wasp's nest-building, ant's collaboration, bird flocking and fish schooling has caught the attention of researchers for their ability to archive complex tasks and working in coordination, which demonstrate some form of swarm intelligence that researchers have taken inspiration from to design and implement swarm robotics systems.[4]

They also report that social insects were able to accomplish their goals like searching for food, alerting the presence of an enemy, or collaborate to lift heavy objects, without having access to the global state or having a leader to guide them. They were only able to accomplish this by utilizing local interactions and communication, which spread to other members and prompted group-wide cooperation.[4]

## 2.2 Swarm robotics and multi robot systems

The early 1980s are when multi robots systems first gained popularity. As the name suggests, multi robot systems introduce the idea of teamwork in order to complete tasks that are challenging or impossible to complete alone by the robots. Seven topics of study have been identified in this field which includes:

- Biological Inspirations;
- Communication
- Localization, mapping, and exploration;
- Object transport and manipulation;
- Motion coordination;
- Reconfigurable robots.

Swarm robotics is a subfield of multi-robot systems that differs from other multi-robot systems in some ways.[5]

## 2.3 Swarm robotics

Swarm robotics has no one definition because it is a rapidly developing area and new research is constantly being done in it. But we can take this definition from a cited paper. "swarm robotics is the study of how large number of relatively simple physically embodied agents can be designed such that a desired collective behavior

emerges from the local interactions among agents and between the agents and the environment.” [6].

The main characters of swarm robotis are:

- The robots in the swarm must be autonomous.
- The number of robots in the swarm is large.
- Homogeneity is required in robots. A small number is acceptable if not.
- Robots must be incompetent with regard to the primary task they must complete, otherwise, they will fail or perform poorly.
- Robots are limited to local communication and sensing. It makes sure that coordination is spread, making scalability one of the system’s characteristics.[4]

## 2.4 Swarm robotics properties

Swarm robotics have some properties that describe the system’s current condition. Bellow are some of these properties:

**Robustness:** the ability of the swarm to still function even with the loss of some members of the group or the faillure of some parts of the system.

**Scalability:** The ability of the system to perform well on smaller or larger group sizes without impacting the performance of the swarm.

**Flexibility** It is the capability of the swarm to adapt and manage the new changes that occur in the environment

**Autonomus:** Implys that there is no central authority controlling the behavior of the swarm and each individual is independent of the others.

**Local communication** The communication among swarm members is local since they don’t have access to the swarm’s overall state. [7][8]

## 2.5 Domain of application

In this section, we will list some of the significant fields domains where swarm robotics fits in, and can impact in solving the domain problem.

### **2.5.1 Tasks that cover a region**

: Because of the widespread sensing capabilities that the swarm has, swarm robotics is most suited for tackling problems that cover an area of the space, such monitoring the environment of a lake or surveillance of a specific area.[6][8]

### **2.5.2 Search and rescue missions**

In different types of accident or disasters that happen like earthquakes where the human intervention is difficult, swarm robots can be deployed for these type of missions. examples of such robots are: polyobot, swarm bot and M-TRAN.

### **2.5.3 Cleaning of oil speals**

A swarm of robots can reduce the cost and time in such incidents. an exemple of robots that were deployed to such tasks is: Seaswarm, which was developed by the senseable group at MIT.

### **2.5.4 Exploration**

To investigate Mars, swarm robots like Marsbees have been created. the same holds true for the CoCoRo swarm, which was employed for in-depth underwater investigation.

### **2.5.5 Agriculture**

As they can be used to improve the agriculture and monitor the status of crops.

## **2.6 Swarm robotics basic tasks and problems**

SR problems can be broken down into fundamental tasks that the swarm frequently carries out in an effort to accomplish its objective. this tasks are: aggregation,d is-persion,pattern formation, coordinated movement, hole avoidance and foraging[3][4].



### **2.6.1 Aggregation**

Aggregation of the robots is performed in order to accomplish some form or to exchange information. This problem can be easy in a centralized system, but difficult in a decentralized one.

### **2.6.2 Dispersion**

For exploration purposes, sometimes, the swarm must cover a wide range of area without losing the connection between the members, in order to expand the group sensing capabilities.

### **2.6.3 Pattern formation**

Sometimes, the swarm must form some specific patterns like circles, squares or lines in order to lift some objects or traverse some ways corridors.

### **2.6.4 Cordinated movement**

It is making an effort to coordinate the group movements by maintaining the established pattern between the robots.

### **2.6.5 Hole avoidance**

As suggested by the name, the group makes an effort to avoid stepping into holes.

### **2.6.6 Foraging**

The swarm aims to locate objects, pick them up, and position them where needed.

### 3 Deep reinforcement learning

### 3.1 Introduction

In this chapter we will provide an overview about the field of reinforcement learning, including its applications, various techniques, and how it relates to swarm robots. Before that, we will briefly discuss machine learning since it is the foundation of RL before we move on.

### 3.2 Machine learning

Machine learning is a sub field of artificial intelligence, which focuses on developing algorithms and models that make prediction and take decisions without being explicitly programmed to do so based on the input data they receive. Generally speaking, there are three types of machine learning algorithms, supervised algorithms which obtain labeled data and attempt to categorize or anticipate the unknown one. Un-supervised learning algorithms, where the data is unlabeled and it is the job of the ml algorithms to extract hidden patterns in it. And finally, reinforcement learning algorithms where the algorithm learns to make decisions based on trial and error and rewards, by interacting with an environment.

### 3.3 Reinforcement learning

Reinforcement learning algorithms are based on a agent that interact and observe an environment in order to take actions that maximize the reward for a given task.

The goal of the agent is to learn an optimal policy ( $\Pi$ ) which maps state to actions in order to get the best possible action to take in a given situation. This can be achieved by maximizing the expected reward that the agent receive from the environment on each step he takes.[9]

Formally, RL can be described as a Markov Decision Process (MPD) where the future state depends only on the current one. An MPD consists of:

- A set of states  $S$
- A set of actions  $A$
- A transition dynamics  $\rho(S_{t+1}|a, S)$ . which give the probability of being in state  $S_{t+1}$  based on the current state  $S$  and the performed action  $A$ .
- A reward function  $R(S_t, A_t, S_{t+1})$  which outputs a scalar reward  $r_t \in R$ .
- A discount factor  $\gamma \in [0, 1]$  which emphasis either immediate or future rewards.

From the above definition, the policy function  $\Pi$  will a map a given state to a probabilty distribution over actions:  $\Pi : S \longrightarrow \rho(A = a|S)$ . The goal of the agent to find the optimal policy  $\Pi$  which maximize the expected return :

$$\Pi^* = \operatorname{argmax}_{\Pi} E[R|\pi]$$

If the MPD is episodic, then the agent will accumulates a set of reward at the end of each episode which is called a return:

$$R = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$$

Setting  $\gamma < 1$  will ensure the convergence of the return in case of non episodic MPDS.[9].

### 3.4 Q learning

Q-learning is a type of model free reinforcement learning algorithm based on the dynamic programming technique, where the agent tries to maximize it rewards by finding the optimale policy  $\Pi$  in an iterative manner. In order to acheive this, the agent uses a type of equation called value function that gives it the value or the reward of being in a state  $s$  and taking action according to it policy  $\Pi$ .

$$V^{\Pi}(S) = E[r(s, a) + \gamma V^{\Pi}(s')]$$

In order to find the optimal value function, hence finding the optimal policy, the equation must satisfy the bellman optimality condition which states that:

$$V^*(S) = \max_a E[r(s, a) + \gamma V^*(s')]$$

The same goes for the action-value function which gives the return if the agent chooses the action  $a$  and follow the optimale policy thereafter.

$$Q^*(S, a) = E[r(s, a) + \max_a \gamma Q^*(s', a)]$$

[10].

### 3.5 Deep Q learning

The problem with traditional Q-learning or RL algorithms in general, is the inability to work in a high dimensional input states where discretization and hand-crafted features extraction is performed in order to it to work. For this, deep learning was combined with traditional Q-learning in order to overcome these challenges.[11]

Deep Q learning was firstly introduced in an article by the deepmind group, where it was tested in atari games in which the agent was given raw input images of the emulator and it was the goal of the agents to win the games by using the Q learning algorithm.

Besides using the bellmen equation to converge and calculate the loss, an experience replay memory was used to store the the experiences of the agents during the game and then sampled randomly to be fed to the neural network to learn and in order to break the correlation that appears when feeding them sequential data.[11]

---

**Algorithm 1** Deep Q-learning with Experience Replay

---

```

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights
for episode = 1, M do
    Initialise sequence  $s_1 = \{x_1\}$  and preprocessed sequenced  $\phi_1 = \phi(s_1)$ 
    for  $t = 1, T$  do
        With probability  $\epsilon$  select a random action  $a_t$ 
        otherwise select  $a_t = \max_a Q^*(\phi(s_t, a; \theta))$ 
        Execute action  $a_t$  in emulator and observe reward  $r_t$  and image  $x_{t+1}$ 
        Set  $s_{t+1} = s_t, a_t, x_{t+1}$  and preprocess  $\phi_{t+1} = \phi(s_{t+1})$ 
        Store transition  $(\phi_t, a_t, r_t, \phi_{t+1})$  in  $D$ 
        Sample random minibatch of transitions  $(\phi_t, a_t, r_t, \phi_{t+1})$  in  $D$ 
        set
            
$$y_j = \begin{cases} r_j, & \text{for terminal } \phi_{j+1}. \\ r_j + \gamma \max_a Q(\phi_{j+1}, a'; \theta), & \text{for non-terminal } Q(\phi_{j+1}). \end{cases} \quad (1)$$

        Perform a gradient descent step on  $(y_j - Q(\phi_j, a_j; \theta))$ 
    end for
end for

```

---

algo1: -the leader robot or first robot will calculate the points and choose one and give it to hi neighbor -the neighbor will do the same and choose one and pass the array to the next.

## **4 Problem formulation And Solution Conception**

## 4.1 Introduction

In this section, we'll go into detail about the objective that needs to be accomplished by our robots, the difficulties they must overcome, and then our suggested solution.

## 4.2 Goal definition

: Our objective in this thesis is to make multi robots form geometric patterns like lines, circles, squares, and triangles while avoiding collisions and moving toward their destination in a quick and efficient manner.

## 4.3 Problem Formulation

### 4.3.1 Environement

We will describe our multi robot environemnt as an MDP where  $S$  are the set of states that each robot will reactive.  $A$  are the set of actions that it can take (left,right,towards ...). and  $R$  are the set of Rewards that each robot will get during each episode. The goal of each robot is to maximize his return by reaching the goal target and avoiding collisons with the wall and other robots. . We will use the DDPG alrorthm to controll the robots in a contius manner

For the environemnt, we will Consider a scenario in which there are  $n$  ( $n > 2$ ) robots moving in a 2D environment. Assuming that every robot is aware of a central point that they must form shapes around or along it.



Figure 1: formation

We'll utilize the polar coordinates  $(R, \Phi)$ , where  $R$  and  $\Phi$  are the radius and angle from the central point, respectively. After assigning each robot a goal position, we will convert it to homogeneous coordinates so it may be reached by them. The desired shapes will be formed by defining the distance and the angle from the center of the formation and by using each robot id to generate unique goal for it.

### 4.3.2 Shape generation

In order to form the different shapes, each robot will have an  $id = (1...n)$  where  $n$  is the number of robots, and he will get assigned a goal based on it using the radius and the distance from the central point. So if the target shape is a triangle the target goals will be as follow:

$$d_i = R \text{ and } \Phi_i = \Phi$$

where  $d_i$  and  $\Phi_i$  are the distance and the angle from the central point respectively.

So if they want to form triangles, squares, or circles this will be their positions:

$$d_i = R$$

$$\Phi_i = 2\pi/id$$

For lines

$$d_i = R * id \quad \Phi_i = 0$$

and we can keep modifying the distance and angle for each robot to get any shape.

And to simplify the job for the robot, we will transform the coordinates from polar to homogenous in order to calculate the distance between the robot and their target goal.

## 4.4 State Representation And Reward Design

### 4.4.1 Introduction

In order to achieve the target shape formation, state and reward are critical components of RL as their representation will affect directly the performance of our robot and will determine the neural network architecture.

### 4.4.2 State Representation

## 4.5 Roboting Operating system

We will use the roboting operating system (ROS), a collection of open source software and tools, to control the robots. ROS will simplify our work by giving us the necessary tools for controlling the robots, determining their positions and angles, gathering sensor data and preparing it for manipulation, and also giving the robots a way to communicate with one another.

### 4.5.1 Ros Basics

There are five concepts that ROS is built on that will allow us to implement our solution in a simple and modular way.



- **Nodes:** ros is made up of nodes where each node is responsible for a single task like moving the robot, getting sensor data, doing some processing ... nodes can communicate between them using topics and services.

- **Topics:** topics are a way for nodes to communicate between them. Each node can either subscribe to a topic thus receiving information from other nodes, or publish to some topic to send information to other nodes.

For example: a node can subscribe to a topic that publishes the position of the robot from another node which is responsible for providing this information.

- **Services:** They are the same as topics but they differ in that data is received when requested by a client in contrast to topics where there is a stream of data updated continuously.

For example: We can have a service that generates goals or resets the simulation.

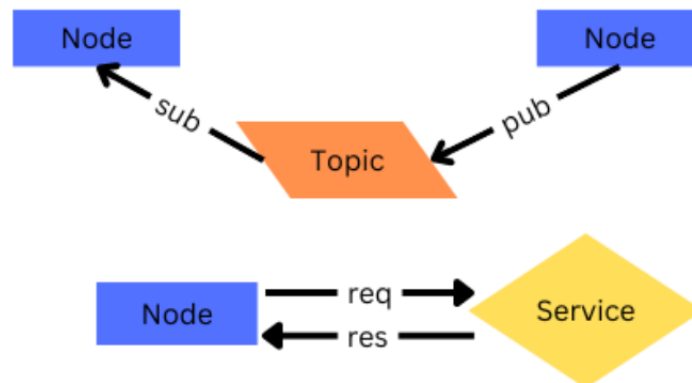


Figure 2: Ros

- 4.6 State Of the art
- 4.7 DQN
- 4.8 DDPG
- 4.9 Thier usages in multi agent pros and cons
- 4.10 problem formulation and solution
- 4.11 we opt for maddpg decentrilzed excution
- 4.12 we opt for maddpg for better perfomance and scalability
- 4.13 ROS and gazebo
- 4.14 Conception of the solution
- 4.15 Rewards desgin
- 4.16 Networks Desgin
- 4.17 talk about how you choose the hyperparameters
- 4.18 how you calculeted tha angle and how lds works
- 4.19 implemtation of the solution

## References

- [1] F. Rubio, F. Valero, and C. Llopis-Albert, “A review of mobile robots: Concepts, methods, theoretical framework, and applications,” *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419839596, 2019.
- [2] L. Bayındır, “A review of swarm robotics tasks,” *Neurocomputing*, vol. 172, pp. 292–321, 2016.
- [3] L. Bayindir and E. Şahin, “A review of studies in swarm robotics,” *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 15, no. 2, pp. 115–147, 2007.
- [4] I. Navarro and F. Matía, “An introduction to swarm robotics,” *Isrn robotics*, vol. 2013, pp. 1–10, 2013.
- [5] T. Arai, E. Pagello, L. E. Parker, *et al.*, “Advances in multi-robot systems,” *IEEE Transactions on robotics and automation*, vol. 18, no. 5, pp. 655–661, 2002.
- [6] E. Şahin, “Swarm robotics: From sources of inspiration to domains of application,” in *Swarm Robotics: SAB 2004 International Workshop, Santa Monica, CA, USA, July 17, 2004, Revised Selected Papers 1*, pp. 10–20, Springer, 2005.
- [7] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, “Swarm robotics: a review from the swarm engineering perspective,” *Swarm Intelligence*, vol. 7, pp. 1–41, 2013.
- [8] I. Olaronke, I. Rhoda, I. Gambo, O. Ojerinde, and O. Janet, “A systematic review of swarm robots,” 2020.
- [9] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “A brief survey of deep reinforcement learning,” *arXiv preprint arXiv:1708.05866*, 2017.
- [10] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, pp. 279–292, 1992.
- [11] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.