Proximal Policy Optimization for Formation Control and Obstacle Avoidance in Multi-Agent Systems

Priyam Sadhukhan

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

4.11.2021

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Priyam Sadhukhan

Entitled: Proximal Policy Optimization for Formation Control and Obstacle Avoidance in Multi-Agent Systems

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________ Chair
Dr. S. Hashtrudi Zad

______________________ External Examiner
Dr. W. Lucia

______________________ Examiner
Dr. S. Hashtrudi Zad

______________________ Supervisor
Dr. R. Selmic

Approved by ______________________
Dr. Y. Shayan, Chair
Department of Electrical and Computer Engineering

Date 4.11.2021 ______________________
Dr. M. Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Proximal Policy Optimization for Formation Control and Obstacle Avoidance in Multi-Agent Systems

Priyam Sadhukhan

In this thesis, deep reinforcement learning (DRL) is used for intelligent formation control and obstacle avoidance in multi-agent systems through reward shaping. The objective of this work is to study the application of proximal policy optimization (PPO) algorithm for maneuvering a formation of agents around obstacles. Each agent in the multi-agent system is modeled as a holonomic second-order integrator and the formation is allowed to shrink while maintaining its shape in order to navigate around obstacles and take the geometric centroid of the formation towards the goal. We investigated both angle-based rewards and bearing-based rewards. Experiments were carried out in a two-dimensional simulation environment with different number of agents and multiple obstacles between the formation and the goal. Curriculum learning was used to train the agents in environments with different initializations for the agents, goal and obstacles. Simulation results show the effectiveness of the different reward schemes.

Acknowledgments

I would like to thank my Supervisor Dr. Selmic for his continous guidance and discussing the various problems with me on a regular basis. I also want to thank the members of my research group for sharing their ideas and for providing valuable advice and feedback. I am grateful to my friends and family, especially my parents for giving me their support throughout the entire process.

This work is supported in part by the Faculty Research Support fund from Concordia University, Montreal.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Formation Control with Deep Reinforcement Learning

Multi-agent formation control involving obstacle avoidance is a challenging task. There are many established traditional control methods designed for this purpose, including consensus control (Dong, Yu, Shi, & Zhong, 2015), sliding mode control (Zhang, Zhang, & Han, 2021), adaptive control based on fuzzy logic (Z. Peng, Wang, & Wang, 2018), and neural network control (Lu, Zhang, Shen, & Zhang, 2019). However, in recent years, deep reinforcement learning (DRL) has been successfully applied for safe, adaptive, multi-agent formation control in (Khan et al., 2019), (Yao et al., 2020) and (Y. Zhou et al., 2019). In this thesis, we propose a new method based on DRL for formation control and obstacle avoidance.

1.2 Literature Review

There are several results that are related to our approach, including (Lin, Yang, Zheng, & Cheng, 2019), where multiple agents are taught to navigate around obstacles through centralized training and decentralized execution and (W. Zhao et al., 2020), which trains independent policies for a group of fixed-wing UAVs using proximal policy optimization (PPO) and value function sharing. Local and global maps are generated in (Tan, Fan, Pan, & Manocha, 2020), which are then processed by a convolutional neural network (CNN) for safe navigation of large number of agents using PPO. A trust-region policy optimization (TRPO)-based approach was used in (Mohseni-Kabir, Isele, & Fujimura, 2019) to combine the outputs for each agent's own objective and that of the group for safe and successful navigation. In (Nguyen, Hatua, & Sung, 2019), collision-free navigation of multiple agents was achieved through multi-agent PPO and multi-agent TRPO but without any formation constraints. Another application of DRL-based flocking was presented in (Yan, Bai, Zheng, & Guo, 2020), where PPO was used along with reward shaping to achieve safe navigation.

In (Y. Zhou et al., 2019), a deep deterministic policy gradient (DDPG) was used, along with a momentum term, for optimizing the value function to achieve formation control and obstacle avoidance. The counterfactual advantage function was used in (Hong & Wang, 2019) to train cooperative agents towards similar objectives. Multi-agent deep deterministic policy gradients (MADDPG) was used in (Jiao & Oh, 2019) for multi-agent navigation, where a recurrent neural network (RNN) was used to create vector embeddings from the observation of individual agents. Safety constraints for multi-agent navigation were satisfied in (Khan et al., 2019) by combining velocity obstacle (VO) method with MADDPG. In (Zhu et al., 2020), MADDPG was used, alongside prioritized experience replay (PER), for flocking maintenance and obstacle avoidance. Large scale flocking of a swarm of UAVs using DDPG is presented in (Wang, Wang, & Zhang, 2018) through local communication with its immediate neighbors. A centralized critic is used for cooperative formation control in (Zuo, Han, & Han, 2010) along with hindsight experience replay (HER) to navigate multiple unicycle models to a given goal position. Fast and optimal navigation of multiple agents was achieved in (Semnani, Liu, Everett, de Ruiter, & How, 2020) using a combination of actor-critic and traditional path planning methods.

Deep Q networks (DQN) were used in (X. Zhou, Wu, Zhang, Guo, & Liu, 2019), in which a formation of agents is taught to flexibly navigate around obstacles and in (Oury Diallo & Sugawara, 2020), for formation control of several agents by training them solely on their own local observations. In (Sui, Pu, Yi, & Tan, 2018), double DQN were used for leader-follower formation control and obstacle avoidance. Navigation and collision avoidance for a multi-agent system was achieved in (Jun, Kim, & Lee, 2019) by using the joint vector of actions for each objective.

1.3 Motivation and Contribution

In this work, we propose a DRL-based method for multi-agent formation control that allows to flexibly maintain its shape while navigating around obstacles for an arbitrary number of agents.

The contributions of this thesis are:

i. Angle-based and bearing-based reward functions that allow the agents to loosely maintain the shape of the formation while navigating around obstacles.

ii. Simulation results of our approaches for different number of agents.

iii. Comparison of results for individual and team rewards for the angle-based approach.

iv. Comparison of results for the angle-based and bearing-based approaches involving individual rewards.

In comparison with previous works such as (X. Zhou et al., 2019), (Hong & Wang, 2019), (Y. Zhou et al., 2019), (Lin et al., 2019), (W. Zhao et al., 2020) that use DRL for formation control and achieve multi-agent formation by maintaining a predefined distance between agents or by setting lower and upper bounds on the allowable distance between agents, we address the problem of maintaining the shape of the formation in terms of angles and bearings, while varying the formation size.

Chapter 2

Preliminaries

This chapter provides the relevant information necessary to understand the methods implemented in this thesis.

2.1 Partially Observable Markov Decision Process

A Markov Decision Process (MDP) (Sutton & Barto, 2018) is a framework used to define a reinforcement learning (RL) problem for a single agent which consists of a tuple $< S, A, T, R, \gamma >$, where S is the global state, A is the action space, $T(s_t, a_t, s_{t+1}) = P(s_{t+1} \mid s_t, a_t)$ is the state transition probability for choosing action $a_t \in A$ in state $s_t \in S$ and receiving the immediate reward $r_t \in R$ according to the reward function $R(s_t, a_t, s_{t+1})$ for every time-step t. The discounted return for horizon H, with a discount factor $\gamma \in (0, 1]$, is given by

$$Rd_t = \sum_{t_s=t}^{H} \gamma^{t_s - t} r_{t_s} . \tag{1}$$

This framework is only viable under full observability, where the global state S is available to the agent. A Partially Observable Markov Decision Process (POMDP) (Sutton &

Barto, 2018) is a generalization of an MDP where only the local observation $o_{t+1} \in O$ is available to the agent and it consists of the tuple $< S, A, T, R, \Omega, O, \gamma >$, where $\Omega = P(o_{t+1} \mid s_{t+1}, a_t)$ determines the observation o_{t+1} in state s_{t+1}. In order to deal with the partial observability, a recurrent neural network (RNN) is utilized (Hausknecht & Stone, 2017) that includes a Long Short Term Memory (LSTM) or Gated Recurrent Unit (GRU) layer.

2.2 Actor-Critic Methods

Actor-critic consists of an actor function and a critic or a value function. In the case of DRL, functions are approximated by neural networks. The critic is used to calculate the action value $Q(s,a)$ while the actor generates a policy π_θ. For a given policy π_θ, at time step t, the corresponding action value, advantage, and value functions are respectively given by

$$Q(s_t, a_t) = \mathop{\mathbb{E}}_{a_t \sim \pi_\theta} \left[r_t + \gamma V_\psi(s_{t+1}) \right], \tag{2}$$

$$A(s_t, a_t) = \left[Q(s_t, a_t) - V_\psi(s_t) \right], \tag{3}$$

$$V_\psi(s_t) = \sum_{t_s = t}^{H} \mathop{\mathbb{E}}_{\pi_\theta} \left(\gamma^{t_s - t} r_{t_s} \right), \tag{4}$$

where $V_\psi(s_t)$ is the state value function for a finite horizon H and denotes the expected value of the return Rd_t for state s_t while following policy π_θ. The advantage of taking action a_t in state s_t is $A(s_t, a_t)$, and θ, ψ are the parameters of the actor and critic networks respectively. The actor network is then updated based on the action value and the state value using the following equations

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{H} \nabla_\theta log \pi_\theta(a_t \mid s_t)(Q(s_t, a_t) - V_\psi(s_t)) \right], \tag{5}$$

$$\theta \leftarrow \theta + \alpha_\theta \nabla_\theta J(\pi_\theta) \tag{6}$$

where α_θ is the learning rate of the actor network. The critic is updated through regression by minimizing the following mean squared error

$$K(V_\psi) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \Big[\sum_{t=0}^{H} (\widehat{R}_t - V_\psi(s_t))^2 \Big] , \tag{7}$$

$$\psi \leftarrow \psi - \beta_\psi \nabla_\psi K(V_\psi) , \tag{8}$$

where, β_ψ is the learning rate of the critic network and $\widehat{R}_t$ is the value target.

2.3 λ-return and Generalized Advantage Estimation

The value target $\widehat{R}_t$ can be a n-step return given by

$$\widehat{R}_t^{(n)} = \sum_{i=0}^{n-1} (\gamma^i r_{t+i} + \gamma^n V_\psi(s_{t+n})) , \tag{9}$$

where n acts as a trade-off between the bias and variance present in the return (X. B. Peng, Abbeel, Levine, & van de Panne, 2018). The output of $V_\psi(s_{t+n})$ is biased during learning. A greater value of n results in a smaller value of the coefficient γ^n at the cost of a higher variance in the return. This is due to multiple terms containing r_t, which may have random values due to stochasticity of the policy and environment dynamics.

A better method for achieving this trade-off is the λ-return (Sutton & Barto, 2018) which is given by

$$\widehat{R}_t^{(\lambda)} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} \widehat{R}_t^{(n)} , \tag{10}$$

where λ is a decay parameter for the exponentially weighted average of n-step returns.

Using the λ-return, (Schulman, Moritz, Levine, Jordan, & Abbeel, 2018) introduced Generalized Advantage Estimation (GAE) in which the advantage is calculated as

$$\widehat{A}_t^{(\lambda)}(s,a) = \sum_{i=0}^{\infty} (\gamma\lambda)^i \delta_{t+i}\,, \tag{11}$$

$$\delta_t = r_t + \gamma V_\psi(s_{t+1}) - V_\psi(s_t)\,, \tag{12}$$

where δ_t is the Temporal Difference (TD) error.

2.4 Proximal Policy Optimization and KL-Divergence

The PPO algorithm (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) is used to learn a stable stochastic policy π_θ for continuous control tasks. It uses multiple steps of stochastic gradient descent (SGD) to perform the following policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathop{\mathbb{E}}_{\tau\sim\pi_\theta} \left[L(s,a,\theta_k,\theta)\right], \tag{13}$$

$$\begin{aligned} L(s,a,\theta_k,\theta) = min\Big(&\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)} A_{\pi_{\theta_k}}, \\ &clip\Big(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}, 1-\varepsilon, 1+\varepsilon\Big) A_{\pi_{\theta_k}}\Big)\,, \end{aligned} \tag{14}$$

where ε is a clipping parameter as described in Spinning Up in DeepRL (Achiam, 2018) by OpenAI. This encourages π_θ to stay close to the current policy π_{θ_k} during the update process.

The KL-Divergence between the current policy π_{θ_k} and the updated policy π_θ is given by

$$\overline{D}_{KL}(\pi_{\theta_k}, \pi_\theta) = \mathop{\mathbb{E}}_{s\sim\pi_{\theta_k}} \left[D_{KL}(\pi_{\theta_k}(\cdot \mid s) \,||\, \pi_\theta(\cdot \mid s))\right]. \tag{15}$$

2.5 Non-stationarity and Parameter Sharing

In case of multi-agent RL, where each agent has its own policy, non-stationarity is introduced during the training process as each agent learns its individual policy, which may not be identical to the others policies. This violates the requirements of an MDP and can be rectified with parameter sharing, which can take multiple forms. In this thesis, we utilize parameter sharing and use a single common policy for all the homogeneous agents.

2.6 Curriculum Learning and Unit Bearing Vector

Curriculum Learning (Yao et al., 2020) is an effective method for training a policy to perform a complicated task by decomposing it into easier sub-tasks. The agent is then trained on each sub-task which gradually increases the difficulty of its overall objective.

The unit bearing vector (S. Zhao & Zelazo, 2017) for an agent i, directed from the agent i to the agent j, is given by

$$\boldsymbol{g}_j^i \triangleq \frac{\boldsymbol{p}^j - \boldsymbol{p}^i}{\|\boldsymbol{p}^j - \boldsymbol{p}^i\|}, \tag{16}$$

where, $\boldsymbol{p}^i$ and $\boldsymbol{p}^j$ are the positions of agents i and j, respectively.

Chapter 3

Problem Formulation

In this chapter, we formulate the problem and model the agents used in simulation experiments in subsequent chapters.

3.1 Problem Formulation

A network of n homogeneous agents is considered, with agents modeled using the second-order dynamics. We frame the problem of navigating the team of agents while maintaining the shape of their formation as a POMDP. At each time-step t, the n agents produce the joint action $a_t = (a_t^1, a_t^2, ..., a_t^n)$ after receiving their individual local observation $o_t = (o_t^1, o_t^2, ..., o_t^n)$, which is computed from the common policy $\pi_\theta : o_t \times a_t \to [0,1]$. The individual actions a_t^i are generated by sampling from the distribution defined by π_θ, where θ represents the policy parameters. Considering the reward function $R : S \times A \to \mathbb{R}$ and the discount factor γ, the expected discounted return for the multi-agent system over a finite horizon H is given by

$$J(\pi) = \mathop{\mathbb{E}}_{\tau \sim p_{\pi_\theta}(\tau)} \big(\sum_{t=0}^{H} \gamma^t R(s_t, a_t) \big) , \tag{17}$$

where, $\tau = (s_0, o_0, a_0, ..., s_H, o_H, a_H)$ is the trajectory of the team and $p_{\pi_\theta}(\tau)$ is its distribution, obtained by following policy π_θ under the state transition distribution $T(s_t, a_t, s_{t+1})$ and observation model $\Omega(s_t, o_t)$. Our objective is to learn the optimal policy π_{θ^*} given by

$$\begin{aligned} &\pi_{\theta^*} = \arg\max_{\pi_\theta} J(\pi_\theta)\,, \\ &s.t. \quad p_{\pi_\theta}(\tau) = p(s_0)p(o_0 \mid s_0)\pi_\theta(a_0 \mid o_0) \\ &\prod_{t=0}^{H-1}[p(s_{t+1} \mid s_t, a_t)p(o_{t+1} \mid s_{t+1})\pi_\theta(a_{t+1} \mid o_{t+1})]\,, \end{aligned} \tag{18}$$

where, the parameters $p(s_0)$, $p(o_0 \mid s_0)$, $p(s_{t+1} \mid s_t, a_t)$, $p(o_{t+1} \mid s_{t+1})$ are all equal to 1 since the environment and the robot dynamics are both deterministic and without any noise.

3.1.1 Modeling the Agents

Agent dynamics: We model n agents as homogeneous robots with identical masses and second-order dynamics. They can traverse the two-dimensional map by applying forces along the x-axis and y-axis at each time step. Each robot is represented as a body of mass m whose kinematic model is given by a second-order integrator

$$\begin{bmatrix} \dot{\boldsymbol{p}} \\ \ddot{\boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{p} \\ \dot{\boldsymbol{p}} \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} \boldsymbol{F}, \tag{19}$$

where $\boldsymbol{p} = (\boldsymbol{x}, \boldsymbol{y})$ is the coordinate of the robot and $\boldsymbol{F} = (\boldsymbol{F}_x, \boldsymbol{F}_y)$ is the force applied by the robot along the corresponding axes. The maximum velocity of the robot is v_{max}.

The second-order model is a natural choice for common robotic systems where the control input is the actuation force. The methods presented in this thesis will also be effective for other dynamics such as first-order models as well as damped models, and will require a separate policy to be trained for the different dynamics.

Sensors: Each robot is equipped with an omnidirectional 1D distance sensor that can separately detect obstacles and other robots along with their approximate distances within a sensing range l_s. Each robot's sensor measurements consist of $b \times d$ distance measurements grouped into b bins, where each bin has a sensor density of d. This is achieved by rotating an array of d sensors spaced $\theta_s = \left(\frac{360}{b \times d}\right)$ degrees apart at a predefined angular velocity ω_s.

Each robot also has access to its own velocity and acceleration measurements along each axis, and equipped with a sensor for a positioning system, either local or global.

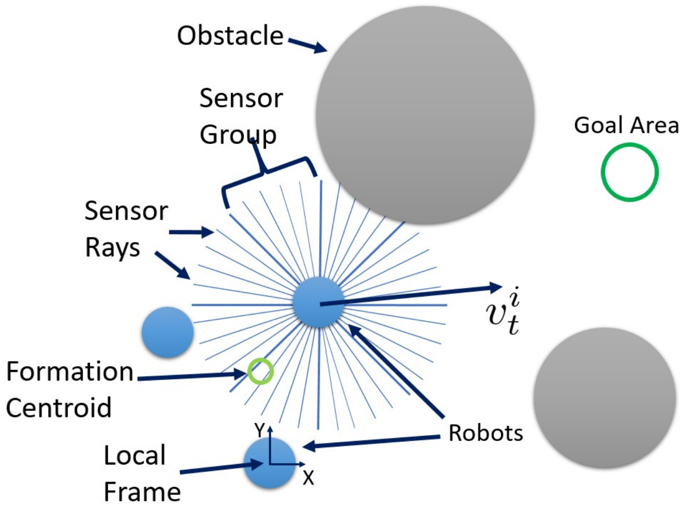


Figure 3.1: Robot formation moving around obstacles

Communication: The communication topology for n robots can be represented as an undirected graph G, which consists of a pair (V,E), where $V = \{1,2,\ldots,n\}$ is the set of vertices and $E \subset (V \times V)$ is the set of undirected edges such that a vertex pair $(i,j) \in E$ implies $(j,i) \in E$. The neighborhood of vertex i is $N^i = \{j \in V \mid (i,j) \in E\}$. It is assumed that the communication graph is static and connected.

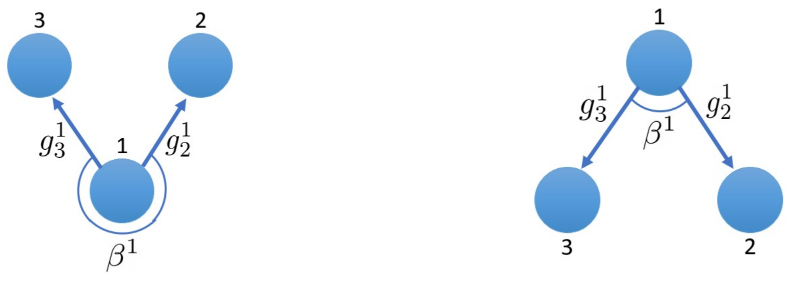
In this thesis, we assume $n(N^i) \geq 2$ for every agent since that is required to maintain a desired angle. We also define $N_2^i \subseteq N^i$, where N_2^i consists of exactly two neighbors that are predefined for each agent during formation initialization. In the case where $n(N^i) > 2$, the neighborhood specifications must not violate the feasibility of maintaning an agent's desired angle or bearing vectors to its two predefined neighbors in N_2^i.

Each agent also maintains its own local coordinate frame and calculates the distance and direction to its respective neighbors, along with the centroid of the formation.

3.1.2 Modeling the Objectives

The agents are required to maintain the shape of their given formation as they maneuver the formation centroid around obstacles as illustrated in Fig. 3.1. The desired orientation of the agents is arbitrarily chosen as anti-clockwise and shown in Fig. 3.2. The objectives are specified below.

Angle-based formation: The n robots are required to keep a desired formation where robot i must maintain a desired angle β^{ref} between the bearing vectors from itself to its two adjacent neighbors in N_2^i, i.e. $\tilde{\beta}^i = \beta^{ref} - \beta^i$, where β^i is the measured angle and β^{ref} is its desired value.



(a) Correct orientation (b) Incorrect orientation

Figure 3.2: Orientation of formation

Bearing-based formation: Alternatively, the n robots can also keep the desired formation with a bearing-based approach, where robot i must maintain a desired bearing $\boldsymbol{g}_j^{i_{ref}}$ towards each of its two adjacent neighbors in N_2^i, i.e. $\tilde{\boldsymbol{g}}_j^i = \boldsymbol{g}_j^{i_{ref}} - \boldsymbol{g}_j^i$, where $\boldsymbol{g}_j^i$ is the measured bearing and $\boldsymbol{g}_j^{i_{ref}}$ is its desired value.

Goal reaching: The goal is reached when $\|\boldsymbol{p}^g - \boldsymbol{p}^c\| \leq \eta_g$, where $\boldsymbol{p}^c$ and $\boldsymbol{p}^g$ are the positions of the formation centroid and goal respectively, and η_g is the radius of the goal area. The centroid of the formation is given by

$$\boldsymbol{p}^c = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{p}^i. \tag{20}$$

3.1.3 Modeling the System Constraints

Obstacle avoidance: Each robot i must be able to maintain a suitably small safe distance η_o from the surface of any obstacle, including other agents, such that $d_o^i \geq d_{safe}$, where $d_o^i = \|\boldsymbol{p}^i - \boldsymbol{p}^o\|, \forall \boldsymbol{p}^o \in O$, where O is the set of all points on the surface of all obstacles within sensing range of robot i. Each agent is penalized for coming in contact with an obstacle or a non-neighboring agent but the simulation is not stopped.

Communication range: Each robot must stay within communication range of its two neighbors, i.e. $d_j^i < d^{ref} < d^{comm}, \forall j \in N^i$, where d^{ref} is the reference distance and d^{comm} is the communication range. Each agent is penalized proportionally to the positive error $max(d_j^i - d^{ref}, 0)$, and when $d_j^i \geq d^{comm}$, communication between agents is still maintained within the simulation.

3.1.4 Observation of an Individual Agent

Each agent's observation o_t^i consists of its own distance measurements l_t^i, which includes the distances towards all obstacles and agents excluding its neighbors, i.e. $j \notin N^i$ with separate channels for obstacles and agents. In addition to these, every agent has access to the following values (z_t^i) along its x-axis and y-axis: its own velocity $\boldsymbol{v}_t^i$, relative distance and velocity to its neighbors $\boldsymbol{d}_{jt}^{iv}, \boldsymbol{v}_{jt}^i, \forall j \in N^i$, relative distance to goal $\boldsymbol{d}_{gt}^{iv}$, relative distance of the formation centroid to goal $\boldsymbol{c}_{gt}^v$, along with its per-step change $(\boldsymbol{c}_{gt}^v - \boldsymbol{c}_{g(t-1)}^v)$. Each agent is also given the distances $d_{jt}^i, \forall j \in N^i$ and its distance to the centroid of the formation d_{ct}^i.

Error measurements e_t^i are provided, including the reference distance error from each of its neighbors $\tilde{d}^i_{jt} = d^i_{jt} - d^{ref}, \forall j \in N^i$. The distance error of an agent from the centroid of the formation $(\eta_c - d^i_{ct})$ is given as well. The angular error $\tilde{\beta}^i_t$ is also provided with respect to the bearing vectors towards its two designated neighbors. The angular errors $\tilde{\beta}^j_t$, $\forall j \in N^i_2$ of its two neighbors are also provided.

Discrete binary values f^i_t that act as flags are also given, including whether the agent is in the correct orientation with its two neighbors and whether all the agents are within the specified reference distance from their respective neighbors.

Lastly, each agent is provided with s^i_t, which includes its previous actions, reward and the current time-step of the simulation. Continuous input values are standardized without subtracting the mean by keeping a running standard deviation for each using Welford's parallel algorithm (Chan, Golub, & LeVeque, 1982).

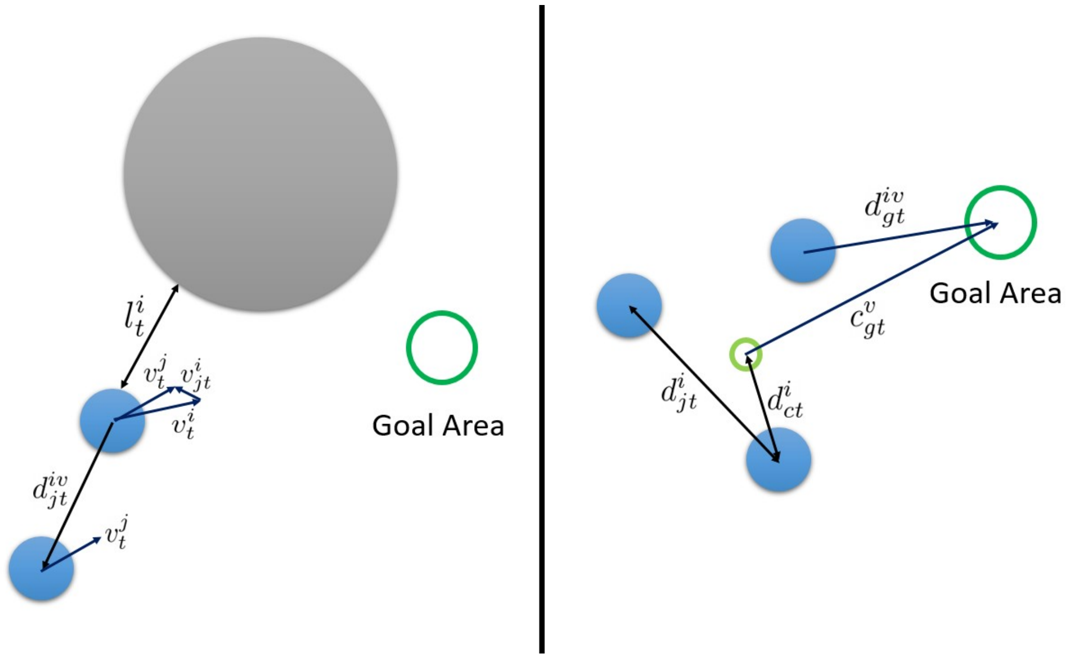


Figure 3.3: Agent obsevations

The agent's observations are also shown in Fig. 3.2, Fig. 3.3 and Table 3.1.

Observation	Definition
l_t^i	Distance measurements to obstacles and non-neighboring agents
$\boldsymbol{v}_t^i$	Velocity components
$\boldsymbol{d}_{jt}^{iv}$	Distance components to neighbor
$\boldsymbol{v}_{jt}^i$	Relative velocity components to neighbor
$\boldsymbol{d}_{gt}^{iv}$	Distance components to goal
$\boldsymbol{c}_{gt}^v$	Distance components of formation centroid to goal
$(\boldsymbol{c}_{gt}^v - \boldsymbol{c}_{g(t-1)}^v)$	Per-step change of $\boldsymbol{c}_g^v$
d_{jt}^i	Distance to neighbor
d_{ct}^i	Distance to formation centroid
$\tilde{d}_{jt}^i$	Reference distance error from neighbor
$(\eta_c - d_{ct}^i)$	Reference distance error from formation centroid
$\tilde{\beta}_t^i$	Angular error with respect to bearing vectors towards two designated neighbors
$\tilde{\beta}_t^j$	Angular error of neighbor
f_t^i	Binary flags indicating correct orientation and positive reference distance error from neighbor
s_t^i	Current time-step t and a_{t-1}^i, r_{t-1}^i

Table 3.1: Agent observations

Chapter 4

Angle Based Reward

In this chapter, we provide the details of our first method and the corresponding results. We do not use a RNN here since each agent only interacts with its neighboring agents and some static obstacles. It is a POMDP but there is no change within the environment outside the agent's observation.

4.1 Methodology

We train a single policy that is shared by the homogeneous agents. Following (Lin et al., 2019), we construct the team level policy from the joint actions of individual agents, i.e. $\pi_\theta^T(o_t) = [\pi_\theta(o_t^1), \pi_\theta(o_t^2), ..., \pi_\theta(o_t^n)]$. However we use two different approaches for constructing the reward function, the first involves rewarding each agent for its own action, while the second one rewards each agent for the actions of the team as a whole. For this method, each agent's observation vector $\boldsymbol{o}_t^i = [l_t^i, \boldsymbol{z}_t^i, e_t^i, s_t^i, f_t^i]$.

4.1.1 Encoding Desired Specifications as Reward Functions

Rewarding each agent for its own actions

We consider v_{max} as the maximum velocity of each agent and define the following reward functions

$$
\begin{aligned}
{}^{angle}r_t^i &= -\frac{\|\beta^f - \beta^i_{t+1}\|}{m_{ang}}\,, &(21)\\
{}^{ref}r_t^i &= -\frac{1}{n(N^i)}\sum_{j\in N^i}\frac{max(d^i_{j(t+1)} - d^{ref}, 0)}{m_{ref}}\,, &(22)\\
{}^{goal}r_t &= \begin{cases} r_{reached}, & \text{if } c_{g(t+1)} \le \eta_g \\ k_1 + \frac{c_{gt} - c_{g(t+1)}}{\Delta t . v_{max}}, & \text{otherwise}\,, \end{cases} &(23)\\
{}^{coll}r_t^i &= \begin{cases} r_j, & \text{if } d^i_{j(t+1)} \le 2R_a\,, \forall j \ne i \\ r_o, & \text{if } d^i_{o(t+1)} < d_{safe} \\ 0, & \text{otherwise}\,, \end{cases} &(24)\\
{}^{move}r_t^i &= -\arccos\Big(\frac{< v^i_t, v^i_{t+1} >}{\|v^i_t\|\|v^i_{t+1}\|}\Big)\,, &(25)
\end{aligned}
$$

where, R_a is the radius of an agent and $\beta^f, m_{ang}, d^{ref}, m_{ref}, r_{reached}, r_j, r_o, d_{safe}, \Delta t, v_{max}$ are constants. Eqn. (25), as given in (Lin et al., 2019), encourages the agents to generate smooth trajectories by penalizing sharp turns.

Then at every time-step, the reward function r_t^i for each agent can be expressed as

$$
\begin{aligned}
r_t^i = K_{angle}\,{}^{angle}r_t^i + K_{ref}\,{}^{ref}r_{ti}^i + K_{goal}\,{}^{goal}r_t \\
+ K_{move}\,{}^{move}r_t^i + K_{coll}\,{}^{coll}r_t^i\,, i = 1, 2, ..., n\,,
\end{aligned} \qquad (26)
$$

where, $K_{angle}, K_{ref}, K_{goal}, K_{move}, K_{coll}$ are constants.

Rewarding each agent for the actions of the team

We consider the average of the individual rewards for each agent as the reward of the team and define the following

$$
\begin{aligned}
{}^{goal}r_t^T &= {}^{goal}r_t \,, &(27)\\
{}^{angle}r_t^T &= avg\left({}^{angle}r_t^i \mid \forall i\right), &(28)\\
{}^{ref}r_t^T &= avg\left({}^{ref}r_t^i \mid \forall i\right), &(29)\\
{}^{move}r_t^T &= avg\left({}^{move}r_t^i \mid \forall i\right), &(30)\\
{}^{coll}r_t^T &= \begin{cases} r_j, & \text{if } d^i_{j(t+1)} \le 2R_a \,, j \ne i \,, \forall i \\ r_o, & \text{if } d^i_{o(t+1)} < d_{safe} \,, \forall i \\ 0, & \text{otherwise}\,. \end{cases} &(31)
\end{aligned}
$$

Then at every time-step, the reward function r_t^i for each agent can be expressed as

$$
\begin{aligned}
r_t^i = K_{angle} {}^{angle}r_t^T + K_{range} {}^{range}r_t^T{}_i + K_{goal} {}^{goal}r_t^T \\
+ K_{move} {}^{move}r_t^T + K_{coll} {}^{coll}r_t^T \,, i = 1, 2, ..., n \,.
\end{aligned} \qquad (32)
$$

4.1.2 Neural Network Structure

The Actor and Critic used are two separate neural networks for function approximation. The Actor outputs two values, which are the means μ_x, μ_y for x-axis and y-axis respectively. The output forces $\boldsymbol{F}_x, \boldsymbol{F}_y$ are obtained by sampling from the two independent Gaussian distributions produced from μ_x, μ_y and a fixed standard deviation σ. Orthogonal initialization is used for all layers.

- Actor network: The Actor network has the structure given in Fig. 4.1. It has 3 layers with 512 neurons in the first layer, 256 neurons in the second layer and 2 outputs.

All the layers have Tanh activation function. The output of the last layer is scaled by 0.01 when using reward scheme defined in section 4.1.1.

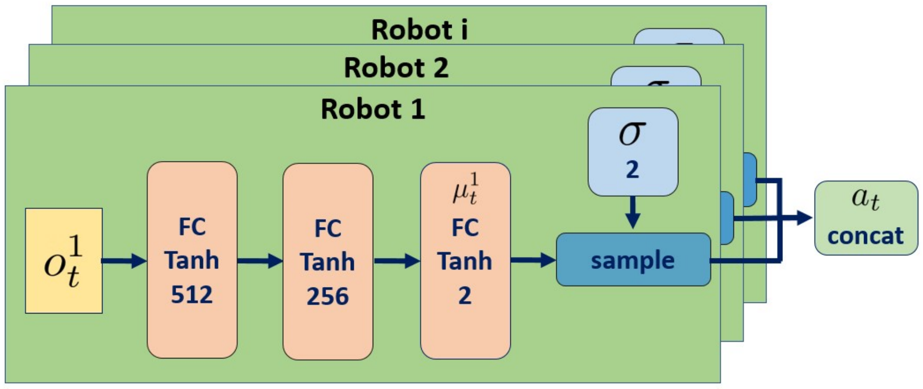


Figure 4.1: Actor network

- Critic network for individual value function: For the policy trained with the individual reward funciton, the Critic network has the strucutre given in Fig. 4.2. It has 4 layers with 512 neurons and Tanh activation function in the first three layers and 1 output with linear activation.

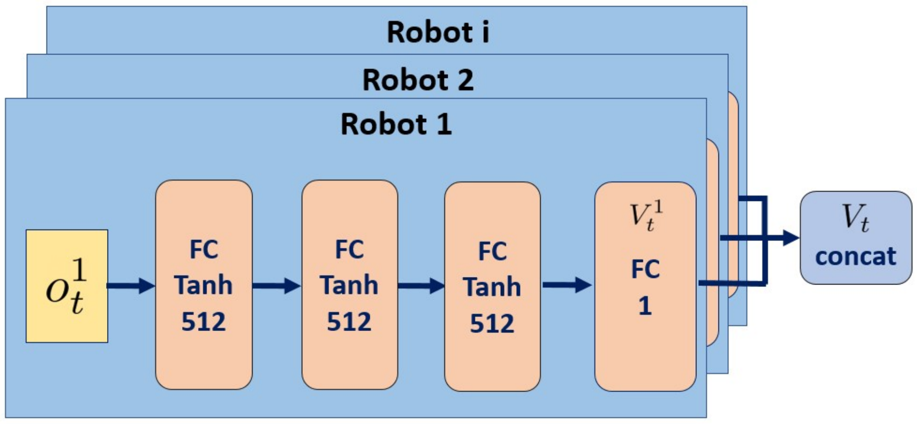


Figure 4.2: Individual critic network

- Critic network for team value function: For the policy trained with the team reward funciton, the Critic network has the strucutre given in Fig. 4.3. It has 5 layers with 512 neurons and Tanh activation function in the first four layers and 1 output with linear activation. The output of the second layer is averaged for each agent and passed through the next three layers.

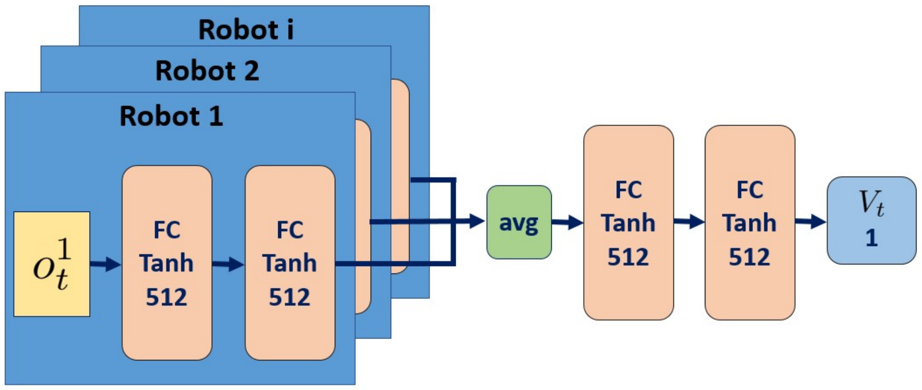


Figure 4.3: Team critic network

4.2 Simulation Setup

For the simulation setup, the Pygame and Pymunk libraries are used to create the simulation environment for the multi-agent system. Pymunk is a 2D rigid body pysics library. The Rllib module (Liang et al., 2018) of the Ray library along with the Pytorch library are used to train the multiple RL agents within a containerized Linux-based Singularity environment. The simulation environment is a 2D graphical game world in $\mathbb{R}^2$, created in Python 3 with a collection of stationary obstacles and the n agents to be controlled. For this method, we consider $n = 3$. The goal area is shown as a large green ring shown in Fig. 3.1. Each pixel width is taken to be $1cm$.

The hardware setup consits of a high performance computing (HPC) cluster that can run multiple experiments in parallel. The processor used is Intel Xeon Gold 6130 @ 2.1 GHz.

4.3 Training

We use each reward scheme to train four policies with different random seeds to account for the effects of randomness and the results are aggregated. We refer to the policies trained using the two reward schemes given in sections 4.1.1 and 4.1.1 as Policy 1 and Policy 2 respectively. In each instance, all agents share the common policy which is learned using the Actor and Critic framework. The shared Actor-Critic architecture is used to emulate parameter sharing between all the agents and also to counter the non-stationarity in MARL which is usually experienced during decentralized learning of each agent's policy. Every agent receives a partial observation $o_t^i \in s_t^i$ at each time-step t. The time interval of the simulation $\Delta t = 0.3s$. Training is carried out in three stages using curriculum learning. The objective for each stage remains the same where the agents are required to navigate around obstacles to reach a specified goal while maintaining the shape of the formation. Agents are allowed to come in contact with each other.

Team specifications: Each holonomic agent is equipped with a 360° distance measurement sensor of range $1.5m$ with measurements taken $\theta_s = 4^\circ$ apart such that $b = 10, d = 9$ and $\omega_s = 60 rad/s$. The radius R_a of each agent is $0.2m$ with maximum linear velocity $v_{max}^i = 0.05m/s$. The dimensions are $15m \times 10m$ with a few small obstacles in the environment.

Environment specifications: For each training run, the starting positions of the agents and the positions of the goal and obstacles are randomly initialized within their specified regions for each phase. The goal is randomly intialized in $[200, 800]$ along the y-axis and either 400 or 1000 along the x-axis. The y coordinate of the centroid of the formation is initialized at either 200 or 800, whichever is further away from the goal. Training is carried out in 3 phases with different specifications.

(1) Phase 1: The centroid of the formation is initialized with the same x coordinate as the goal. The agents are intialized in random order within the formation and are not

required to maintain a desired angle with their neighbors. Three large obstacles with radius 200 cm are randomly initialized in $[400,700]$ or in $[700,1000]$ along the x-axis, whichever range is futher away from the formation centroid. The y coordinates of the obstacles are randomly initialized in $[100,900]$. One small obstalce with radius 70 cm is also initialized at the mid-point between the formation centroid and the goal.

(2) Phase 2: Similar to Phase 1 but the agents are required to maintain the desired angle with their neighbors.

(3) Phase 3: The centroid of the formation is initialized with the an x coordinate of either 400 or 1000 that is different from the goal. There is random placement of one large obstacle with radius 80 cm and two smaller ones with radius 50 cm in $[400,1000]$ along the x-axis and in $[100,900]$ along the y-axis.

The common policy shared by the agents is trained for 10000 iterations in Phase 1, 10000 iterations in Phase 2 and 20000 iterations in Phase 3. Total training time is around 60 hours wall-clock time.

Reward specifications: The coefficients of the reward function are specified in Table 4.1. Rewards are clipped at every step such that $r_t^i \in [-10,10]$.

PPO and optimizer settings: Generalized Advantage Estimation (GAE) (Schulman et al., 2018) is used for calculating the target value of the Critic with $\lambda = 0.95$. The discount factor $\gamma = 0.995$ and $\sigma = 0.7$. Policy clip parameter $\varepsilon_p = 0.2$ and value function clipping $\varepsilon_v = 20$. Total number of workers is 6 with 2 vectorized environments each and a soft horizon $H_s = 128$ steps means each episode ends when the condition for $r_{reached}$ is satisfied or after every $T_s = 2500$ steps. Training is carried out in mini-batches of size 512 for 10 epochs at the end of each iteration. Early stopping is implemented when $\overline{D}_{KL}(\pi_\theta, \pi_{\theta_k}) \geq 0.03$. The learning rate of the ADAM optimizer is $l_r = 0.5 \times 10^{-4}$.

Coefficient	Value
K_{goal}	0.5
K_{ref}	2
K_{angle}	0.4
K_{move}	2
K_{coll}	1
$r_{reached}$	9
η_g	$0.5m$
m_{ang}	30°
β^f	60°
d^{ref}	$2.4m$
d^{comm}	$3m$
m^{ref}	$0.1m$
r_o	-6
r_j	-1
d_{safe}	$0.02m$

Table 4.1: Coefficient and reward values

4.4 Results and Discussion

The training progress is shown in Fig. 4.4 and Fig. 4.5 for Policy 1 and Policy 2 respectively, with the shaded region representing the standard deviation.

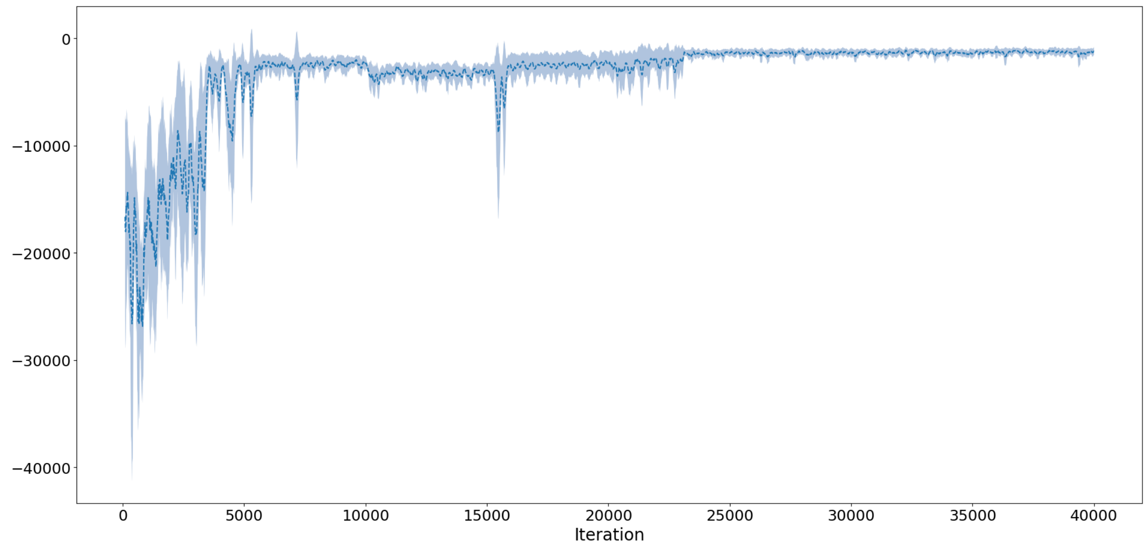


Figure 4.4: Mean reward while training with Policy 1

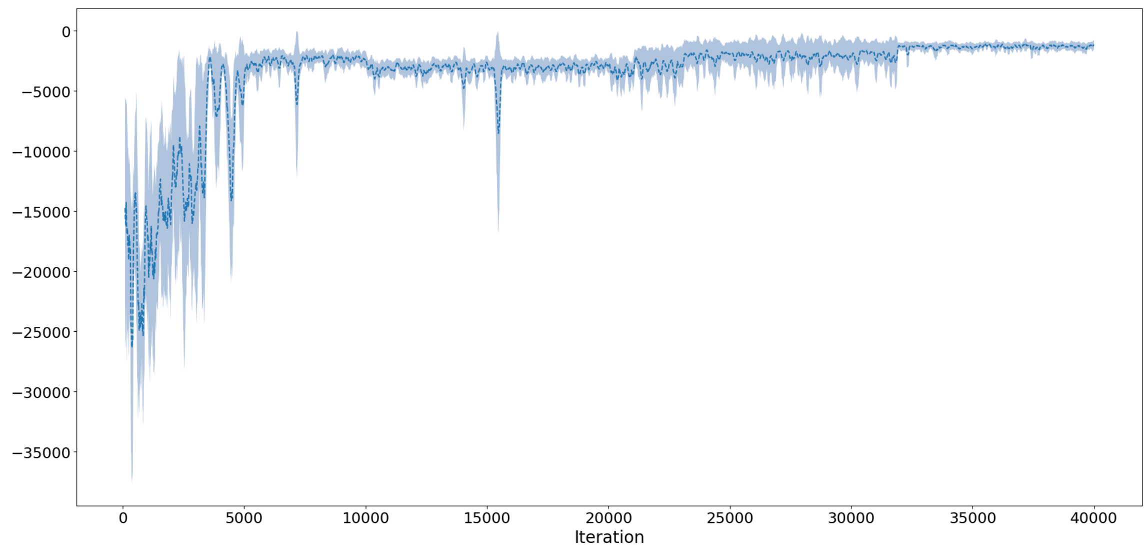


Figure 4.5: Mean reward while training with Policy 2

Evaluation is carried out with each trained policy in a Phase 3 environment 500 times. The fraction of trajectories that successfully reached the goal is 5% and 7% for Policy 1 and Policy 2 respectively. The error values of the successful trajectories are compared in Fig. 4.6 with corresponding means and standard deviations shown as error bars. Policy

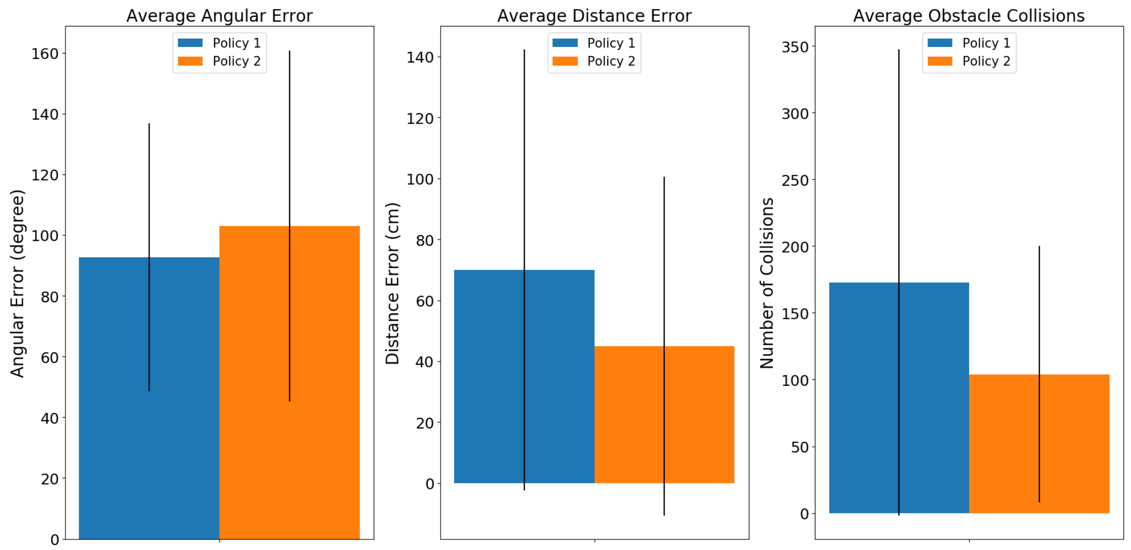


Figure 4.6: Evaluation results with Policy 1 and Policy 2

2 has lesser number of collisions with obstacles and also reduced distance error between

agents due to better coordination as a team. However, it is at the cost of marginal increase in angular error. The average distance error is used in Eqn. (22) and only indicates the positive error. The tolerance limit of the distance error from each agent to its neighbor is $(d^{comm} - d^{ref}) = 60cm$, exceeding which is a violation of the communication constraint. It is given greater importance over the angular error due to our choice of reward coefficients. It is also more important to maintain connection among the agents, at the cost of the formation shape. However, some evaluation runs fail to keep the agents within communication range all the time as can be seen from the corresponding error bar.

There are a substantial number of collisisons during the evaluation phase which reduce the effectiveness of the learned policy. However, the agents have a low momentum and are not greatly affected by the collisions. The obstalce avoidance performance of the policy can be improved by assigning a penalty proportional to the impact velocity of the collision.

An example of a successful trajectory from each policy is shown in Fig. 4.7 and Fig. 4.8. Both policies generate a curved trajectory around obstacles though Policy 2 appears to maintain less distance between agents in Fig 4.8. Agent 1 in Fig. 4.7 suffers a collision with the larger obstacle.

The primary objective of our approach is to maintain the shape of the formation which can be effectively measured by the average angular error of the agents. It can be seen in Fig. 4.9 the average angular error is around 30°.

Both trajectories have no positive distance error as shown in Fig. 4.10. The average errors of the two trajectories in Fig. 4.9 and Fig. 4.10 are shown for different number of steps because each trajectory has a different time of completion as obtained during policy evaluation.

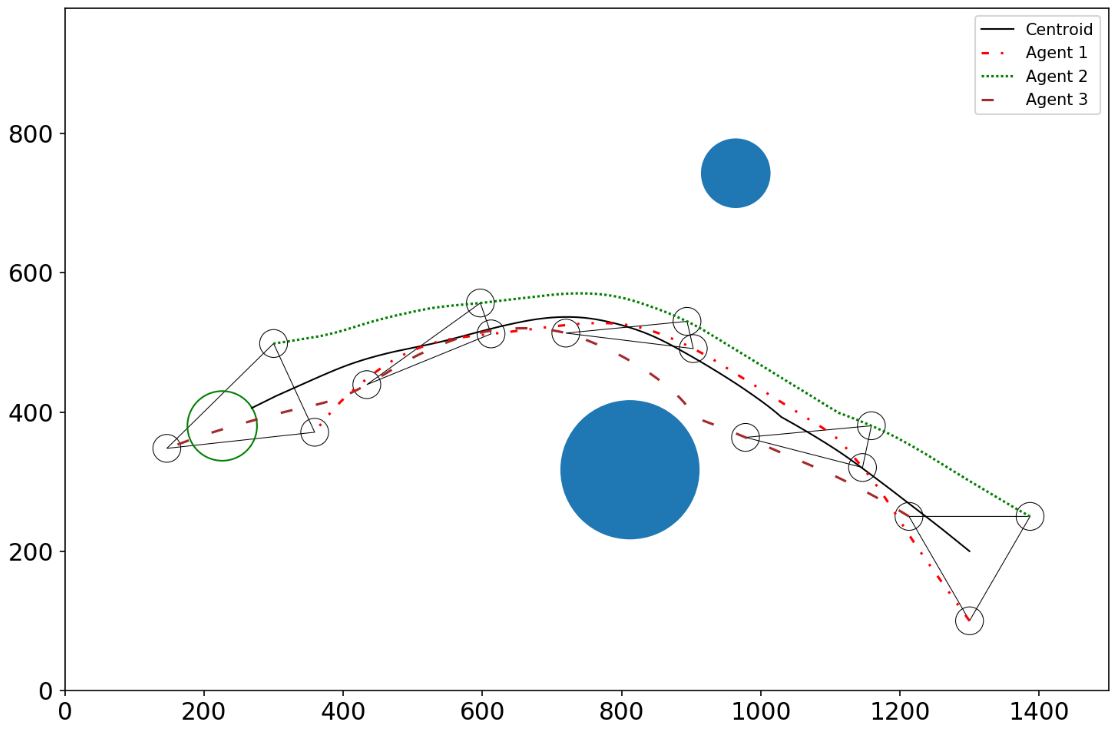


Figure 4.7: Agent trajectories after evaluation with Policy 1

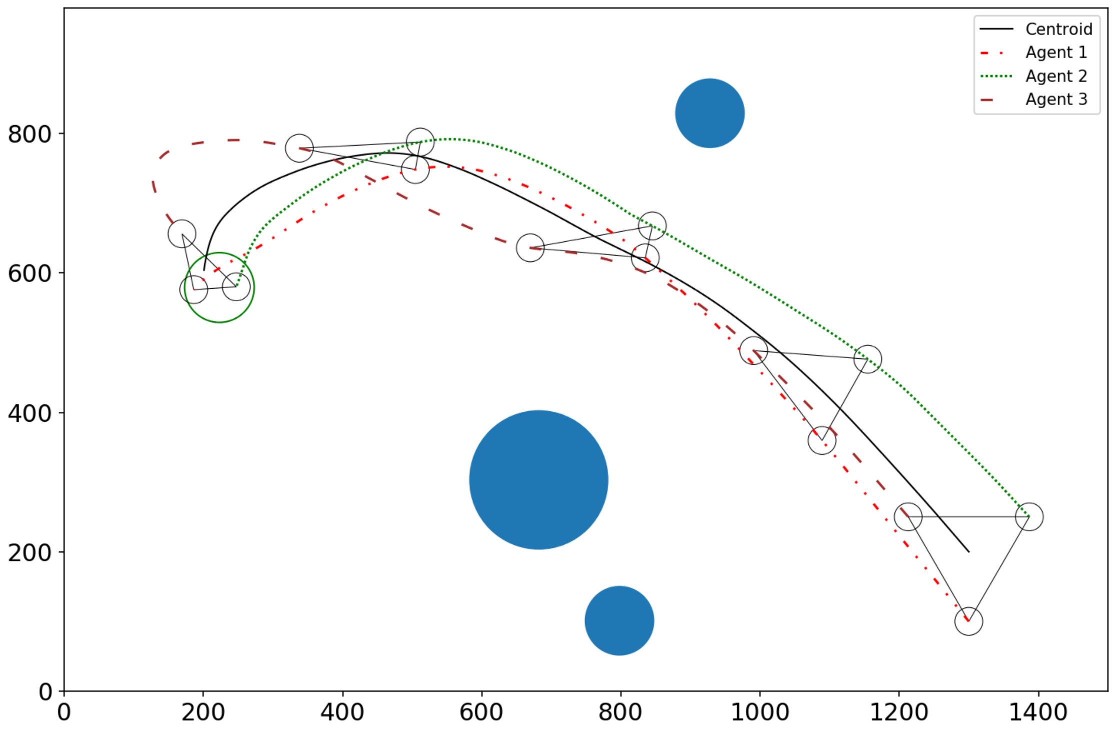


Figure 4.8: Agent trajectories after evaluation with Policy 2

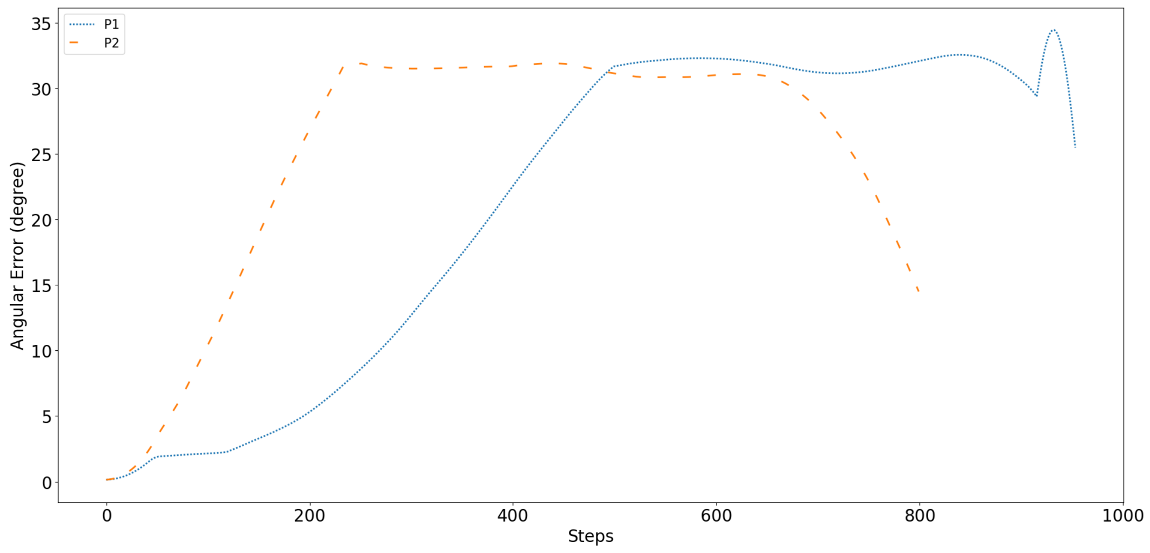


Figure 4.9: Average angular errors for trajectories in Fig. 4.7 and Fig. 4.8

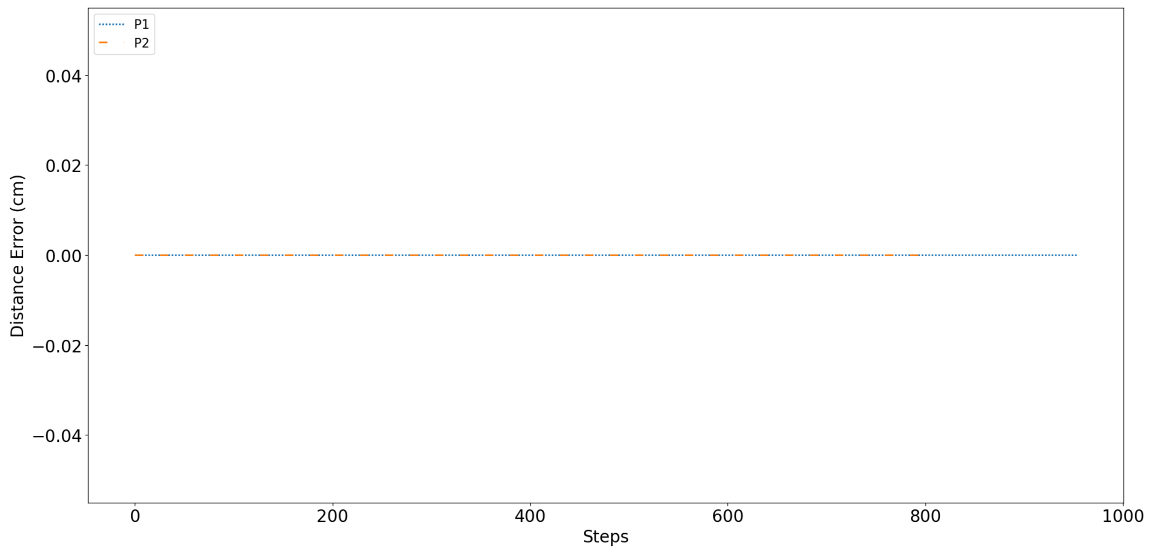


Figure 4.10: Average distance errors for trajectories in Fig. 4.7 and Fig. 4.8

Chapter 5

Bearing Based Reward

In this chapter, we provide the details of our second method and the corresponding results. This method utilizes a RNN since an agent can interact with other non-neighboring agents with whom no information is exchanged besides the agent's own distance measurements. These result in changes within the environment outside the agent's observation since those other agents are non-stationary.

5.1 Methodology

We train a single policy that is shared by the homogeneous agents. Following (Lin et al., 2019), the team level policy is constructed from the joint actions of individual agents, i.e. $\pi_\theta^T(o_t) = [\pi_\theta(o_t^1), \pi_\theta(o_t^2), ..., \pi_\theta(o_t^n)]$. For this method, each agent's observation vector $o_t^i = [s_t^i, z_t^i, e_t^i, l_t^i, f_t^i]$.

5.1.1 Encoding Desired Specifications as Reward Functions

We consider v_{max} as the maximum velocity magnitude of each agent and define the following reward functions for encoding the desired behavior of an agent i and the formation as a whole. In order to encourage the agents to reach the goal area within the desired number of steps, we define

$$
{}^{time}r_t = \begin{cases} r_{late}, & \text{if } (t+1) = T_s \\ 0, & \text{otherwise}, \end{cases} \tag{33}
$$

where, T_s is the number of steps after which the environment is reset and r_{late} is a constant. The reward for maintaining the angle-based formation is given by

$$
{}^{angle}r_t^i = -\frac{\|\beta^{ref} - \beta_{t+1}^i\|}{m_{ang}}\,, \tag{34}
$$

where, β^{ref} is the reference angle and m_{ang} is a constant. The reward for maintaining the bearing-based formation is given by

$$
{}^{bearing}r_t^i = -\frac{0.5}{n(N_2^i)} \sum_{j \in N_2^i} \|\boldsymbol{g}_j^{i_{ref}} - \boldsymbol{g}_{j(t+1)}^i\|\,, \tag{35}
$$

where $\boldsymbol{g}_j^{i_{ref}}$ is the refernece unit bearing vector. In order to keep neighboring agents within a desired range, we define

$$
{}^{dist}r_t^i = -\frac{1}{n(N^i)} \sum_{j \in N^i} \frac{max(d_{j(t+1)}^i - d^{ref}, 0)}{m_{dist}}\,, \tag{36}
$$

where, d^{ref} and m_{dist} are constants. The following function rewards all the agents to stay within a specified distance from the centroid of the formation

$$
{}^{centroid}r_t = \begin{cases} r_c, & \text{if } d_{c(t+1)}^i \leq \eta_c \\ 0, & \text{otherwise}, \end{cases} \tag{37}
$$

where, r_c, η_c are constants. The agents are required to take the centroid of the formation to the goal area, for which we define

$$
{}^{goal}r_t = \begin{cases} r_{reached}, & \text{if } c_{g(t+1)} \leq \eta_g \\ k_1 + \frac{c_{gt} - c_{g(t+1)}}{\Delta t . v_{max}}, & \text{otherwise}, \end{cases} \tag{38}
$$

where, $r_{reached}, k_1, \eta_g, \Delta t$ are constants. For avoiding collisions with obstacles and other agents, we define

$$
{}^{coll}r^i_t = \begin{cases} r_j, & \text{if } d^i_{j(t+1)} \leq 2R_a, \forall j \neq i \\ r_o, & \text{if } d^i_{o(t+1)} < d_{safe} \\ 0, & \text{otherwise}, \end{cases} \tag{39}
$$

where, R_a is the radius of an agent and r_j, r_o, d_{safe} are constants. We also define

$$
{}^{move}r^i_t = -\arccos\Big(\frac{< \boldsymbol{v}^i_t, \boldsymbol{v}^i_{t+1} >}{\|\boldsymbol{v}^i_t\| \|\boldsymbol{v}^i_{t+1}\|}\Big), \tag{40}
$$

which encourages the agents to generate smooth trajectories, (Lin et al., 2019). In order to encourage each agent to keep its movement aligned with its neighbors, we define

$$
{}^{velocity}r^i_t = -\frac{1}{n(N^i)} \sum_{j \in N^i} \arccos\Big(\frac{< \boldsymbol{v}^i_{t+1}, \boldsymbol{v}^j_t >}{\|\boldsymbol{v}^i_{t+1}\| \|\boldsymbol{v}^j_t\|}\Big). \tag{41}
$$

Then at every time-step, the reward function r^i_t for each agent can be expressed as

$$
\begin{aligned} r^i_t = & {}^{time}r^i_t + K_{angle} {}^{angle}r^i_t + K_{bearing} {}^{bearing}r^i_t + K_{dist} {}^{dist}r^i_{ti} \\ & + {}^{centroid}r^i_{ti} + K_{goal} {}^{goal}r_t + K_{move} {}^{move}r^i_t + K_{vel} {}^{velocity}r^i_t \\ & + K_{coll} {}^{coll}r^i_t, \ i = 1, 2, ..., n, \end{aligned} \tag{42}
$$

where, K_{angle}, $K_{bearing}$, K_{dist}, K_{goal}, K_{move}, K_{vel}, K_{coll} are constants.

5.1.2 Neural Network Structure

The Actor and Critic are two separate neural networks that are used for function approximation. The Actor outputs two values, which are the means μ_x, μ_y for x-axis and y-axis, respectively. The output forces $\boldsymbol{F}_x, \boldsymbol{F}_y$ are obtained by sampling from the two independent Gaussian distributions produced from μ_x, μ_y and a fixed standard deviation σ. Orthogonal initialization is used for all layers.

Actor network: The actor network has the structure given in Fig. 5.1. It has 3 layers with 512 neurons in the first layer, 256 neurons in the second recurrent layer, and 2 outputs. All the layers have tanh activation function.

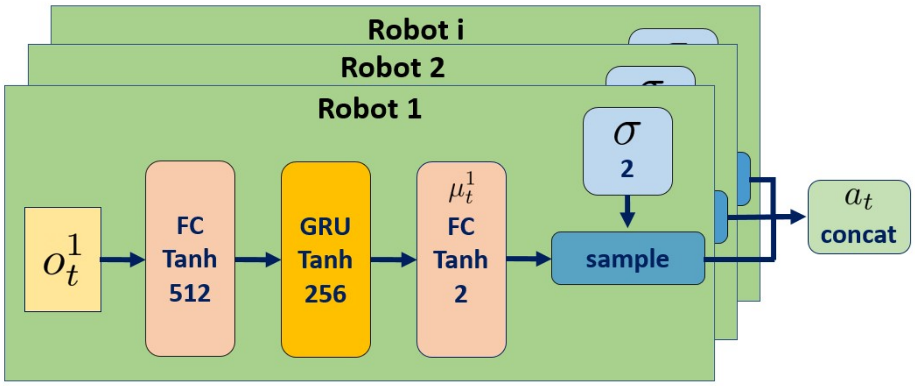


Figure 5.1: Actor network

Critic network: The critic network has the structure given in Fig. 5.2. It has 4 layers with 512 neurons in the first layer, 512 neurons in the second recurrent layer, 256 neurons in the fully connected layer, tanh activation function in the first three layers, and 1 output with linear activation.

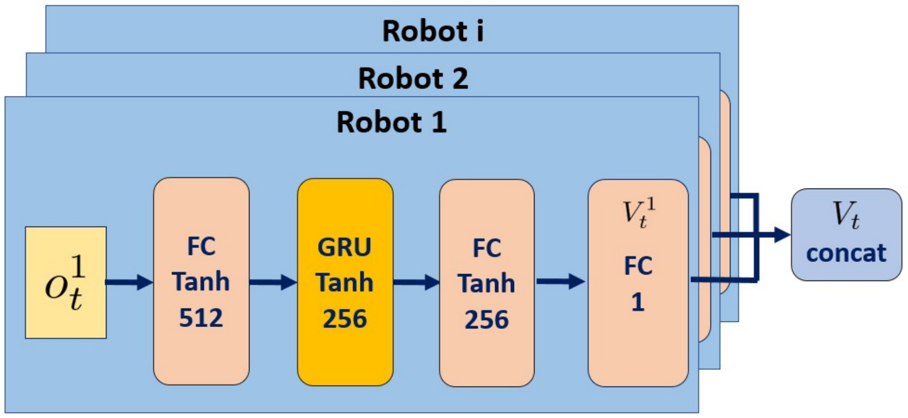


Figure 5.2: Critic network

5.2 Simulation Setup

For the simulation setup, the Pygame and Pymunk libraries are used to create the simulation environment for the multi-agent system. Pymunk is a 2D rigid body pysics library. The Rllib module (Liang et al., 2018) of the Ray library along with the Pytorch library are used to train the multiple RL agents within a containerized Linux-based Singularity environment. The simulation environment is a 2D graphical game world in $\mathbb{R}^2$, created in Python 3 with a collection of stationary obstacles and the n agents to be controlled. For this method, we consider $n = 6$. The goal area is shown as a large green ring shown in Fig. 3.1. Each pixel width is taken to be $1cm$.

The hardware setup consits of a high performance computing (HPC) cluster that can run multiple experiments in parallel. The processor used is Intel Xeon Gold 6130 @ 2.1 GHz.

5.3 Training

The agents share a common policy, which is learned using the Actor and Critic framework. The shared Actor-Critic architecture is used to emulate parameter sharing between all the agents and also to counter the non-stationarity in MARL which is usually experienced during decentralized learning of each agent's policy. Every agent receives a partial observation $o_t^i \in s_t^i$ at every time-step t. The time interval of the simulation is $\Delta t = 0.3s$. Curriculum learning is used to train the agents in 3 separate stages within the same environment.

Team specifications: Each agent is equipped with a 360° distance measurement sensor of range $1.5m$ with measurements taken $\theta_s = 4^\circ$ apart such that $b = 10, d = 9$ and $\omega_s = 60 rad/s$. It can distinguish between obstacles and other agents excluding its designated neighbors. The radius of each agent is $R_a = 0.2m$ and the maximum linear velocity is $v_{max} = 0.05 m/s$.

Environment specifications: The dimensions are $15m \times 10m$ with a few large obstacles between the agents' starting positions and the goal. The agents must learn to avoid collisions with these obstacles and each other. For each training run, the starting positions of the agents and the positions of the goal and obstacles are randomly initialized within their specified regions for each phase. The goal is randomly intialized in $[200, 800]$ along the y-axis and either 400 or 1100 along the x-axis. The y coordinate of the centroid of the formation is initialized at either 200 or 800, whichever is further away from the goal. Training is carried out in 3 stages with different specifications.

(1) Stage 1: The centroid of the formation is initialized with the same x coordinate as the goal. The agents are intialized in random order within the formation and are required to reach the goal while staying within a pre-defined distance from the center of the formation. Three large obstacles with radius 200 cm are randomly initialized in $[400, 750]$ or in $[750, 1100]$ along the x-axis, whichever range is futher away from the formation centroid. The y coordinates of the obstacles are randomly initialized in $[100, 900]$. One small obstalce with radius 70 cm is also initialized at the mid-point between the formation centroid and the goal.

(2) Stage 2: Similar to Phase 1 but each agent now also stays within a pre-defined range of its two neighbors.

(3) Stage 3: The centroid of the formation is initialized with the an x coordinate of either 400 or 1000 that is different from the goal. There is random placement of two large obstacles with radius 80 cm in $[400, 1100]$ along the x-axis and in $[100, 900]$ along the y-axis. Along with the above objectives, the agents are now required to maintain the shape of the formation as they navigate around obstacles.

The common policy shared by the agents is trained for 5000 iterations in Stage 1, 5000 iterations in Stage 2 and 15000 iterations in Stage 3. For each training run, the starting positions of the agents, the positions of the goal as well as the obstacles are randomly initialized. Total training time is around 100 hours of wall-clock time.

Reward specifications: The coefficients of the reward function are specified in Table 5.1. Reward clipping is done to ensure $r_t^i \in [-10, 10]$.

The positions of the 6 agents on the vertices of a regular hexagon is shown in Fig. 5.3. The reference bearing $\boldsymbol{g}_{\boldsymbol{j}}^{i_{ref}}$ for each agent in the 6-agent system is specified in Table 5.2

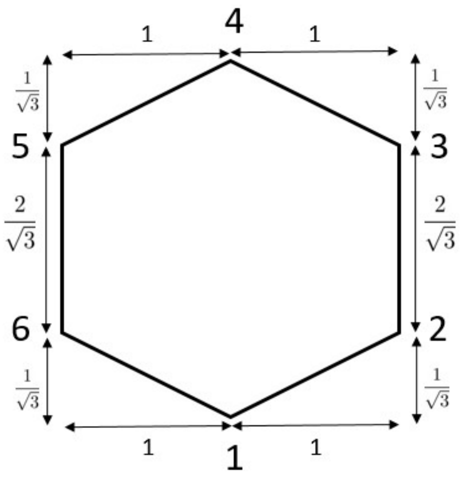


Figure 5.3: Positions of agents on the vertices of a regular hexagon

where $d = \sqrt{3}$.

Coefficient	Value
K_{goal}	0.5
K_{dist}	1
K_{move}	0.3
$K_{velocity}$	0.1
K_{coll}	0.5
r_c	0.01
$r_{reached}$	4
r_{late}	-10
η_g	$0.4m$
η_c	$1.2m$
m_{ang}	30°
β^{ref}	120°
d^{ref}	$1.5m$
d^{comm}	$3m$
m^{dist}	$0.1m$
k_1	-1.1
r_o	-5
r_j	-1
d_{safe}	$0.05m$

Table 5.1: Coefficient and reward values

Reference bearing	Preceeding agent	Succeeding agent
$\boldsymbol{g_j}^{1_{ref}}$	$\left(-1, \frac{1}{d}\right)_{j=6}$	$\left(1, \frac{1}{d}\right)_{j=2}$
$\boldsymbol{g_j}^{2_{ref}}$	$\left(-1, -\frac{1}{d}\right)_{j=1}$	$\left(0, \frac{2}{d}\right)_{j=3}$
$\boldsymbol{g_j}^{3_{ref}}$	$\left(0, -\frac{2}{d}\right)_{j=2}$	$\left(-1, \frac{1}{d}\right)_{j=4}$
$\boldsymbol{g_j}^{4_{ref}}$	$\left(1, -\frac{1}{d}\right)_{j=3}$	$\left(-1, -\frac{1}{d}\right)_{j=5}$
$\boldsymbol{g_j}^{5_{ref}}$	$\left(1, \frac{1}{d}\right)_{j=4}$	$\left(0, -\frac{2}{d}\right)_{j=6}$
$\boldsymbol{g_j}^{6_{ref}}$	$\left(0, \frac{2}{d}\right)_{j=5}$	$\left(1, -\frac{1}{d}\right)_{j=1}$

Table 5.2: Reference bearings to neighbors

We set the values of $K_{angle} = 0.3$, $K_{bearing} = 0$, while training with the angle-based reward for Policy 1 and set $K_{bearing} = 0.3$, $K_{angle} = 0$ while training with the bearing-based reward for Policy 2. For each category, we train 3 policies with different random seeds to take into account the effects of randomness. The moving average of the rewards from all the policies during training are shown in Fig. 5.4 and Fig. 5.5 with a rolling window of size 100 along with the standard deviation of the rewards indicated by the shaded regions.

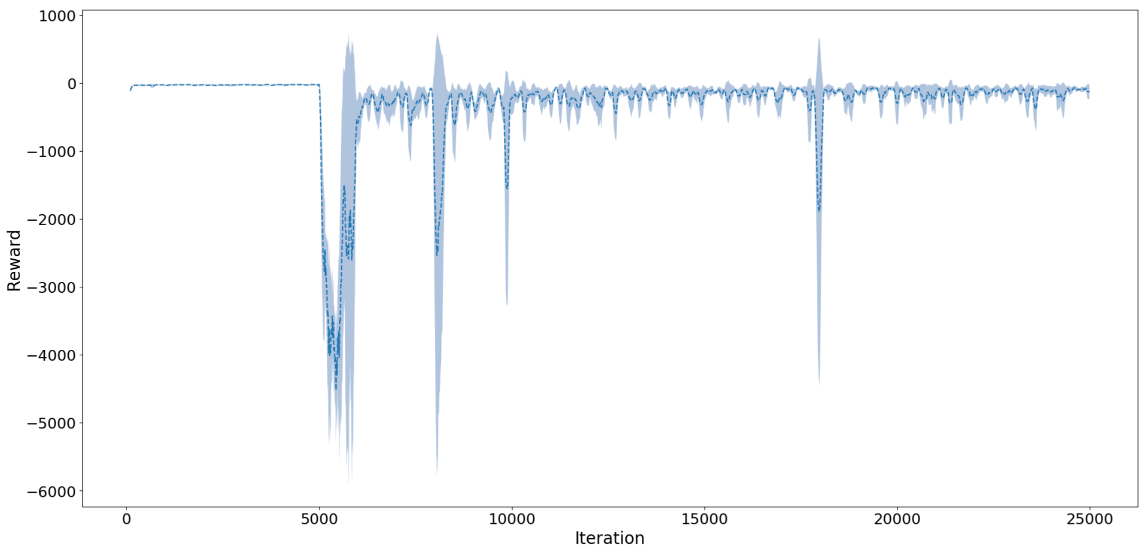


Figure 5.4: Reward during training with Policy 1

PPO and optimizer settings: GAE is used for calculating the target value of the Critic with $\lambda = 0.95$. The discount factors are $\gamma = 0.995$ and $\sigma = 0.7$, policy clip parameter is $\varepsilon_p = 0.2$ and the value function clipping is $\varepsilon_v = 20$. The total number of workers is 7 with 2 vectorized environments each and a soft horizon $H_s = 128$ steps, meaning the environment is not reset at episode end but only when the condition for $r_{reached}$ is satisfied or after every $T_s = 2500$ steps. Training is carried out in mini-batches of size 512 for 10 epochs at the end of each iteration where the size of the training batch after an iteration is $128 \times 7 \times 2$. The learning rate of the ADAM optimizer is $l_r = 5 \times 10^{-5}$. Early stopping is implemented when $\overline{D}_{KL}(\pi_\theta, \pi_{\theta_k}) \geq 0.03$.

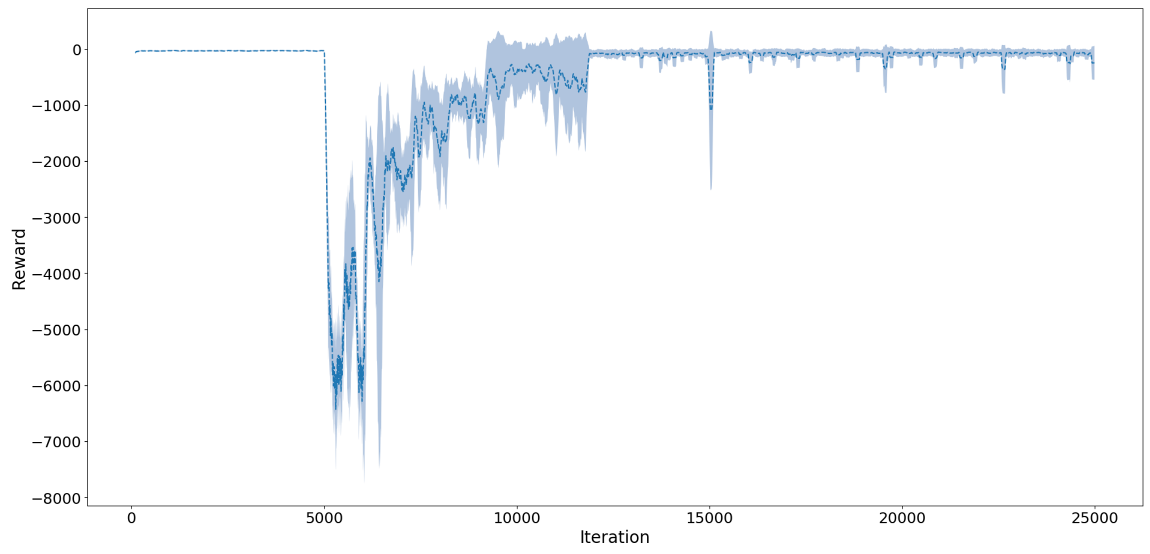


Figure 5.5: Reward during training with Policy 2

5.4 Evaluation and Discussion

Evaluation is carried out for 200 runs with each trained policy from which we show 2 trajectories from each category. We denote the trajectories generated by Policy 1 as A1, A2, and those generated by Policy 2 as B1, B2. The success rate of reaching the goal for Policy 1 is 8% while that for Policy 2 is 11%, where success is defined as the fraction of the trial runs in which the centroid of the formation reached the goal area.

The average errors from the successful trials are given in Fig. 5.6. Policy 2 is better able to maintain the distance specifications compared to Policy 1 with marginal improvement in average angular error.

As can be seen in the Figs. 5.7 - 5.10 depicting agent trajectories, both policies effectively take the agents to the goal region by bringing the centroid of the formation to the goal. However, some of the agents come in contact with the obstacles. In Fig. 5.7 especially, the formation grows as it moves by the obstacle because there is no negative penalty unless there is a collision. We also do not enforce any distance requirements so long as each agent stays within a distance of d^{ref} from its corresponding neighbors and r_c from the

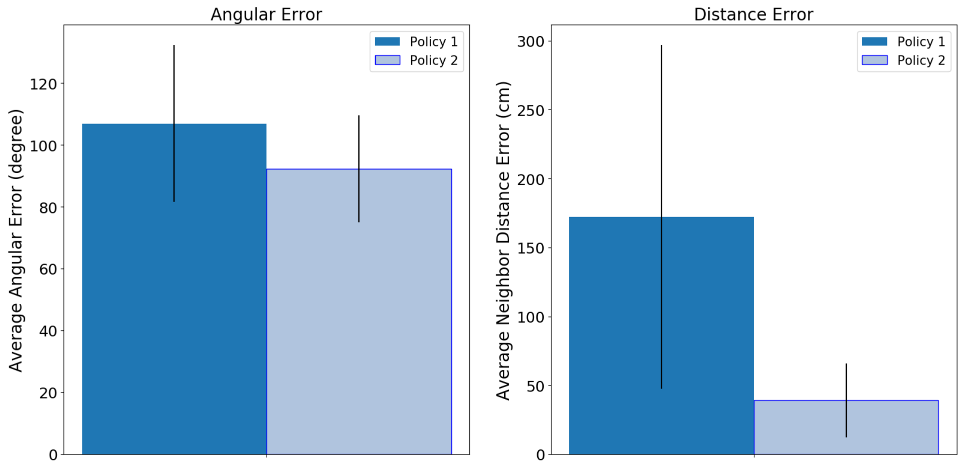


Figure 5.6: Average evaluation errors for the two policies

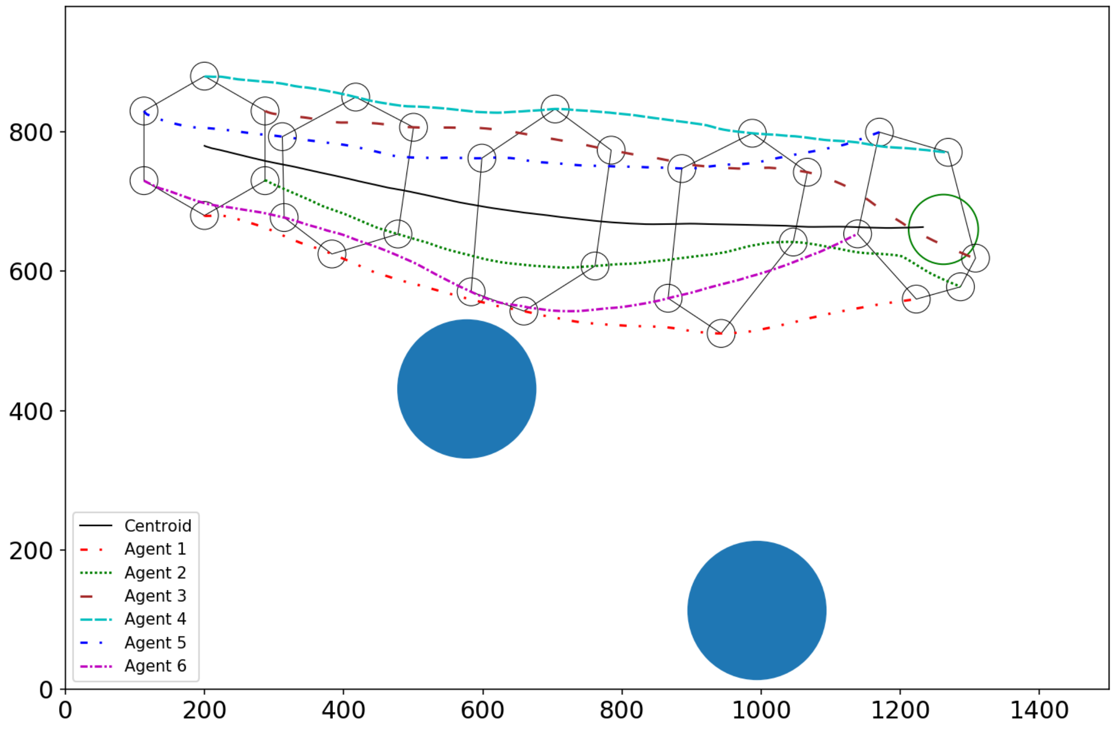


Figure 5.7: Trajectory A1 after evaluation with Policy 1

formation centroid.

The average angular errors in Fig. 5.11 show that the trajectories for both policies

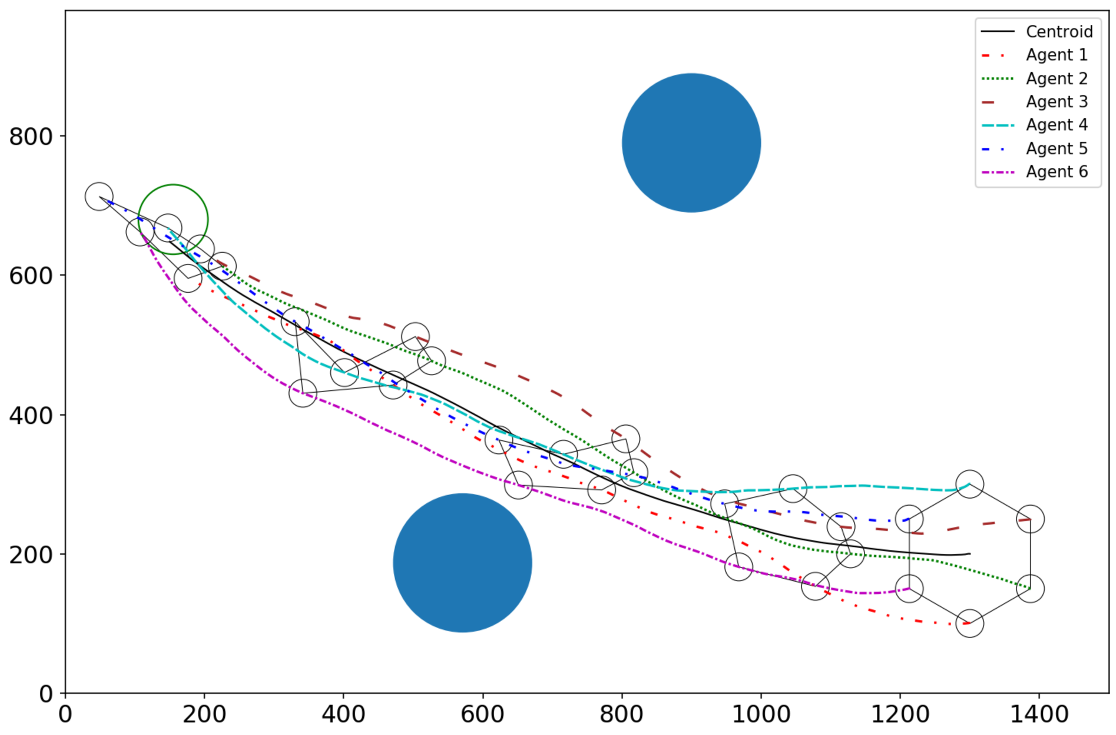


Figure 5.8: Trajectory A2 after evaluation with Policy 1

undergo a steady increase in angular error. The error is greater in places where the agents fail to maintain the shape of the formation and come too close together. For trajectories A2 and B1, the error appears to grow even as the formation reaches the goal since the agents receive a large positive reward on completion which outweighs the negative reward from the angular error.

The average distance error shown in Fig. 5.12 for the trajectories is not significantly high since the agents all manage to stay close together throughout their trajectories. The average distance error is used in Eqn. 36 and only indicates the positive error. The tolerance limit of the distance error from each agent to its neighbor is $(d^{comm} - d^{ref}) = 150cm$ exceeding which is a violation of the communication constraint.

The average errors of the four trajectories in Fig. 5.11 and Fig. 5.12 are shown for different number of steps because each trajectory has a different time of completion as obtained during policy evaluation.

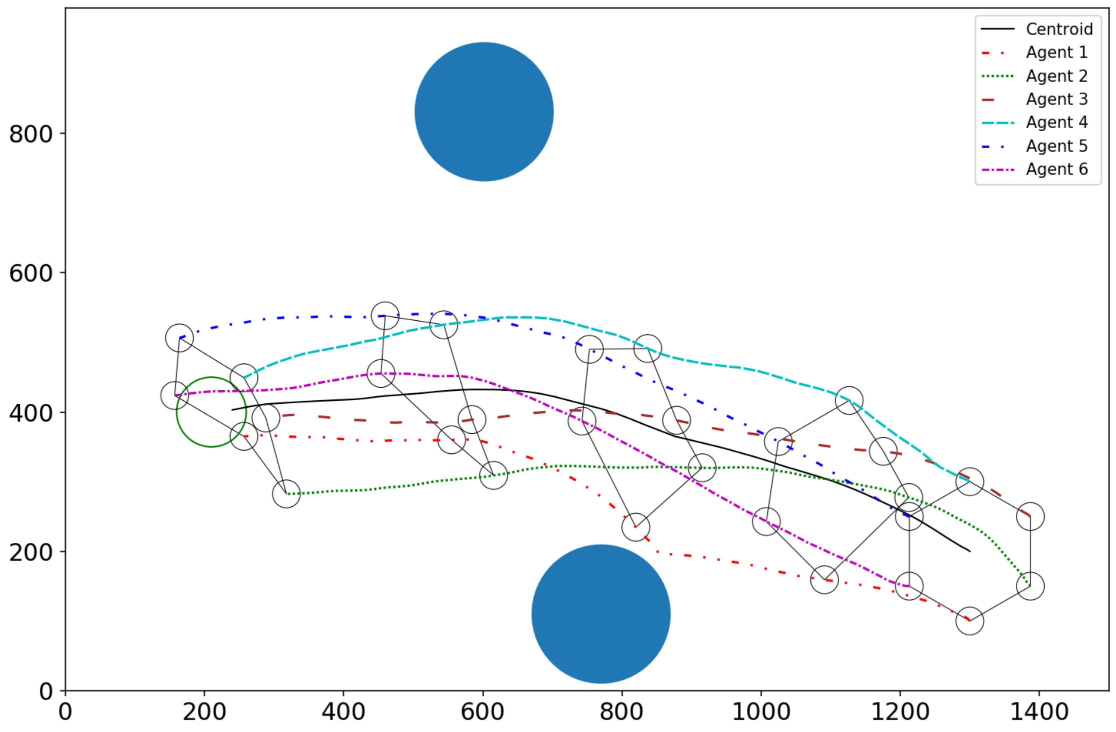


Figure 5.9: Trajectory B1 after evaluation with Policy 2

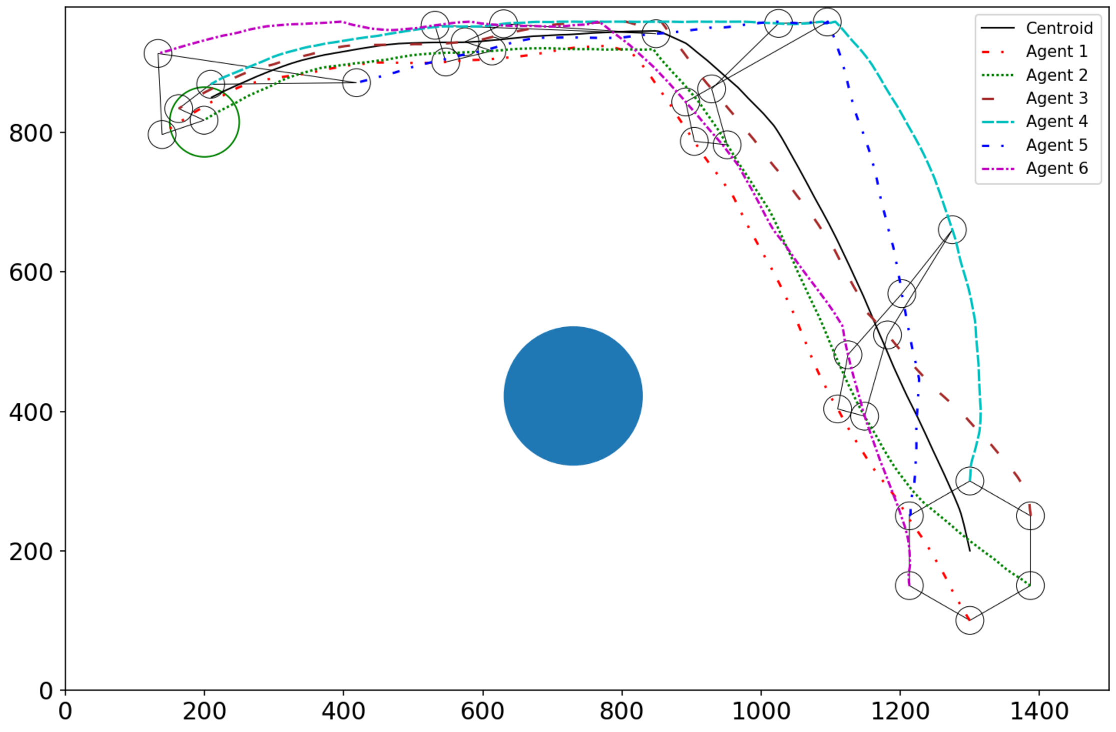


Figure 5.10: Trajectory B2 after evaluation with Policy 2

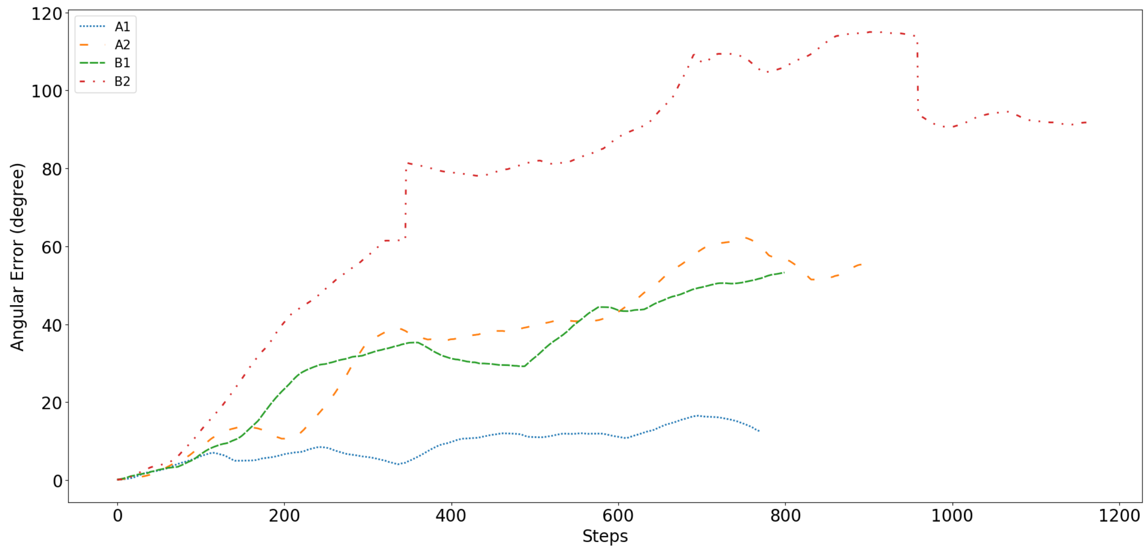


Figure 5.11: Average angular errors for the trajectories in Figs. 5.7 - 5.10

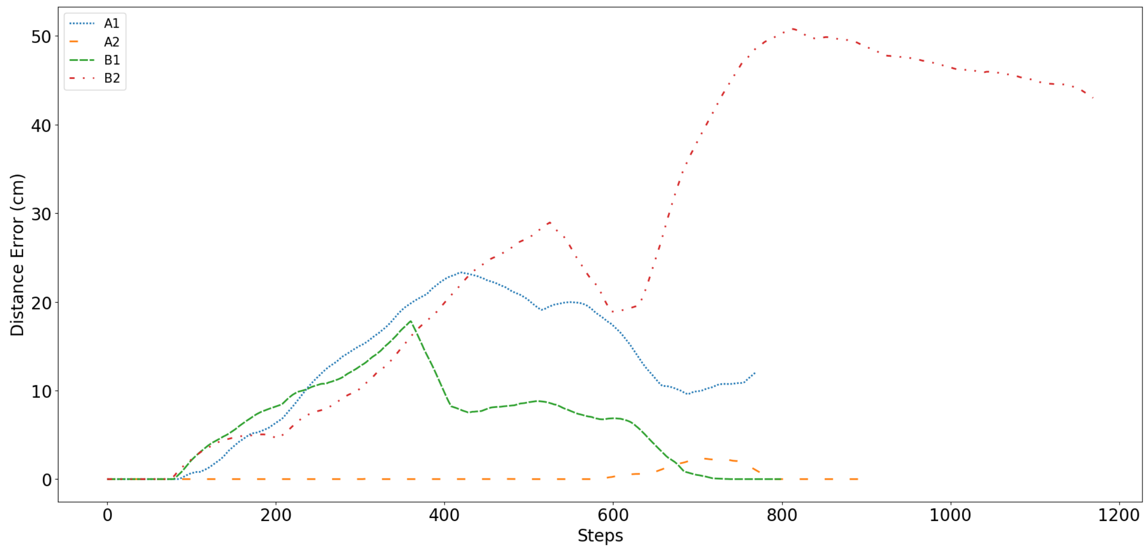


Figure 5.12: Average distance errors for the trajectories in Figs. 5.7 - 5.10

Chapter 6

Conclusion and Future Work

In this thesis, we explored different approaches using PPO for maintaining the formation shape of a group of second-order, holonomic agents using angle-based error and bearing-based error in the shaped reward function. The different methods presented here can effectively navigate the centroid of the formation to the goal, while keeping a low average distance error between the agents and their corresponding neighbors.

The low success rates in comparision to other related works using DRL such as (Lin et al., 2019), (X. Zhou et al., 2019) and (Y. Zhou et al., 2019) can be explained by the differences in constraint requiremetns and information available to the agents compared to our work. In (Lin et al., 2019), the agents are not required to maintain any formation shape and (Y. Zhou et al., 2019) requires a leader-follower approach and does not have any formation requirements while avoiding obstacles. The formation requirements of (X. Zhou et al., 2019) are similar to our work in that the agents flexibly maintain the shape of the formation while maneuvering around obstalces. However, it does not mention the success rate of the evaluation results and presents the performance of the policy for a few individual trajectories.

Using the team reward results in marginally better performance than indivudual rewards for the angle-based approach. The bearing-based reward also leads to better performance compared to the angle-based reward. However, the shape of the formation gets distorted when the agents come too close to their respective neighbors or fail to maintain their relative orientation.

Future work will involve applying the methods presented in this thesis to non-holonomic agents along with the application of constraint satisfaction for formation shape and obstacle avoidance.

Appendix A

Acronym	Meaning
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
PPO	Proximal Policy Optimization
UAV	Unmanned Aerial Vehicle
CNN	Convolutional Neural Network
TRPO	Trust Region Policy Optimization
DDPG	Deep Deterministic Policy Gradients
MADDPG	Multi Agent Deep Deterministic Policy Gradients
RNN	Recurrent Neural Network
VO	Velocity Obstacle
PER	Prioritized Experience Replay
HER	Hindsight Experience Replay
DQN	Deep Q Networks
MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
GAE	Generalized Advantage Estimation
SGD	Stochastic Gradient Descent

Table A.1: Acronyms

References

Achiam, J. (2018). *Spinning Up in Deep Reinforcement Learning.*

Chan, T. F., Golub, G. H., & LeVeque, R. J. (1982). Updating Formulae and a Pairwise Algorithm for Computing Sample Variances. In H. Caussinus, P. Ettinger, & R. Tomassone (Eds.), *COMPSTAT 1982 5th Symposium held at Toulouse 1982* (pp. 30–41). Heidelberg: Physica-Verlag HD. doi: 10.1007/978-3-642-51461-6_3

Dong, X., Yu, B., Shi, Z., & Zhong, Y. (2015, January). Time-Varying Formation Control for Unmanned Aerial Vehicles: Theories and Applications. *IEEE Trans. Contr. Syst. Technol.*, *23*(1), 340–348. doi: 10.1109/TCST.2014.2314460

Hausknecht, M., & Stone, P. (2017, January). Deep Recurrent Q-Learning for Partially Observable MDPs. *arXiv:1507.06527 [cs].*

Hong, Z., & Wang, Q. (2019, November). Deterministic Policy Gradient Based Formation Control for Multi-Agent Systems. In *2019 Chinese Automation Congress (CAC)* (pp. 4349–4354). Hangzhou, China: IEEE. doi: 10.1109/CAC48633.2019.8996660

Jiao, Z., & Oh, J. (2019, December). End-to-End Reinforcement Learning for Multi-agent Continuous Control. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)* (pp. 535–540). Boca Raton, FL, USA: IEEE. doi: 10.1109/ICMLA.2019.00100

Jun, H. W., Kim, H. J., & Lee, B. H. (2019, May). Goal-Driven Navigation for Non-holonomic Multi-Robot System by Learning Collision. In *2019 International Conference on Robotics and Automation (ICRA)* (pp. 1758–1764). Montreal, QC, Canada: IEEE. doi: 10.1109/ICRA.2019.8793810

Khan, A., Zhang, C., Li, S., Wu, J., Schlotfeldt, B., Tang, S. Y., … Kumar, V. (2019, November). Learning Safe Unlabeled Multi-Robot Planning with Motion Constraints. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 7558–7565). Macau, China: IEEE. doi: 10.1109/IROS40897.2019.8968483

Liang, E., Liaw, R., Moritz, P., Nishihara, R., Fox, R., Goldberg, K., … Stoica, I. (2018, June). RLlib: Abstractions for Distributed Reinforcement Learning. *arXiv:1712.09381 [cs]*.

Lin, J., Yang, X., Zheng, P., & Cheng, H. (2019, August). End-to-end Decentralized Multi-robot Navigation in Unknown Complex Environments via Deep Reinforcement Learning. In *2019 IEEE International Conference on Mechatronics and Automation (ICMA)* (pp. 2493–2500). Tianjin, China: IEEE. doi: 10.1109/ICMA.2019.8816208

Lu, Y., Zhang, C., Shen, T., & Zhang, W. (2019, November). Adaptive Formation Scaling Maneuver Control of Autonomous Surface Vehicles with Uncertain Dynamics and Bearing Constraints. In *2019 Chinese Automation Congress (CAC)* (pp. 128–133). Hangzhou, China: IEEE. doi: 10.1109/CAC48633.2019.8997500

Mohseni-Kabir, A., Isele, D., & Fujimura, K. (2019, May). Interaction-Aware Multi-Agent Reinforcement Learning for Mobile Agents with Individual Goals. In *2019 International Conference on Robotics and Automation (ICRA)* (pp. 3370–3376). Montreal, QC, Canada: IEEE. doi: 10.1109/ICRA.2019.8793721

Nguyen, T. T., Hatua, A., & Sung, A. H. (2019, October). Cumulative Training and Transfer Learning for Multi-Robots Collision-Free Navigation Problems. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)* (pp. 0305–0311). New York City, NY, USA: IEEE. doi: 10.1109/UEMCON47517.2019.8992945

Oury Diallo, E. A., & Sugawara, T. (2020, July). Multi-Agent Pattern Formation: A Distributed Model-Free Deep Reinforcement Learning Approach. In *2020 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). Glasgow, United Kingdom: IEEE. doi: 10.1109/IJCNN48605.2020.9207657

Peng, X. B., Abbeel, P., Levine, S., & van de Panne, M. (2018, August). DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, *37*(4), 1–14. doi: 10.1145/3197517.3201311

Peng, Z., Wang, J., & Wang, D. (2018, May). Distributed Maneuvering of Autonomous Surface Vehicles Based on Neurodynamic Optimization and Fuzzy Approximation. *IEEE Trans. Contr. Syst. Technol.*, *26*(3), 1083–1090. doi: 10.1109/TCST.2017.2699167

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2018, October). High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv:1506.02438 [cs]*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017, August). Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*.

Semnani, S. H., Liu, H., Everett, M., de Ruiter, A., & How, J. P. (2020, April). Multi-Agent Motion Planning for Dense and Dynamic Environments via Deep Reinforcement Learning. *IEEE Robot. Autom. Lett.*, *5*(2), 3221–3226. doi: 10.1109/LRA.2020.2974695

Sui, Z., Pu, Z., Yi, J., & Tan, X. (2018, July). Path Planning of Multiagent Constrained Formation through Deep Reinforcement Learning. In *2018 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). Rio de Janeiro: IEEE. doi: 10.1109/IJCNN.2018.8489066

Sutton, R. S., & Barto, A. (2018). *Reinforcement learning: An introduction* (Second edition ed.). Cambridge, MA London: The MIT Press.

Tan, Q., Fan, T., Pan, J., & Manocha, D. (2020, October). DeepMNavigate: Deep Reinforced Multi-Robot Navigation Unifying Local & Global Collision Avoidance. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 6952–6959). Las Vegas, NV, USA: IEEE. doi: 10.1109/IROS45743.2020.9341805

Wang, C., Wang, J., & Zhang, X. (2018, November). A Deep Reinforcement Learning Approach to Flocking and Navigation of UAVs in Large-Scale Complex Environments. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)* (pp. 1228–1232). Anaheim, CA, USA: IEEE. doi: 10.1109/GlobalSIP.2018.8646428

Yan, P., Bai, C., Zheng, H., & Guo, J. (2020, November). Flocking Control of UAV Swarms with Deep Reinforcement Leaming Approach. In *2020 3rd International Conference on Unmanned Systems (ICUS)* (pp. 592–599). Harbin, China: IEEE. doi: 10.1109/ICUS50048.2020.9274899

Yao, S., Chen, G., Pan, L., Ma, J., Ji, J., & Chen, X. (2020, November). Multi-Robot Collision Avoidance with Map-based Deep Reinforcement Learning. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)* (pp. 532–539). Baltimore, MD, USA: IEEE. doi: 10.1109/ICTAI50040.2020.00088

Zhang, Z., Zhang, K., & Han, Z. (2021). A Novel Cooperative Control System of Multi-Missile Formation Under Uncontrollable Speed. *IEEE Access*, *9*, 9753–9770. doi: 10.1109/ACCESS.2021.3049571

Zhao, S., & Zelazo, D. (2017, September). Translational and Scaling Formation Maneuver Control via a Bearing-Based Approach. *IEEE Trans. Control Netw. Syst.*, *4*(3), 429–438. doi: 10.1109/TCNS.2015.2507547

Zhao, W., Chu, H., Miao, X., Guo, L., Shen, H., Zhu, C., … Liang, D. (2020, August).

Research on the Multiagent Joint Proximal Policy Optimization Algorithm Controlling Cooperative Fixed-Wing UAV Obstacle Avoidance. *Sensors*, *20*(16), 4546. doi: 10.3390/s20164546

Zhou, X., Wu, P., Zhang, H., Guo, W., & Liu, Y. (2019). Learn to Navigate: Cooperative Path Planning for Unmanned Surface Vehicles Using Deep Reinforcement Learning. *IEEE Access*, *7*, 165262–165278. doi: 10.1109/ACCESS.2019.2953326

Zhou, Y., Lu, F., Pu, G., Ma, X., Sun, R., Chen, H.-Y., & Li, X. (2019, November). Adaptive Leader-Follower Formation Control and Obstacle Avoidance via Deep Reinforcement Learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4273–4280). Macau, China: IEEE. doi: 10.1109/IROS40897.2019.8967561

Zhu, P., Dai, W., Yao, W., Ma, J., Zeng, Z., & Lu, H. (2020). Multi-Robot Flocking Control Based on Deep Reinforcement Learning. *IEEE Access*, *8*, 150397–150406. doi: 10.1109/ACCESS.2020.3016951

Zuo, G., Han, J., & Han, G. (2010). Multi-robot Formation Control Using Reinforcement Learning Method. In D. Hutchison et al. (Eds.), *Advances in Swarm Intelligence* (Vol. 6145, pp. 667–674). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-642-13495-1_82