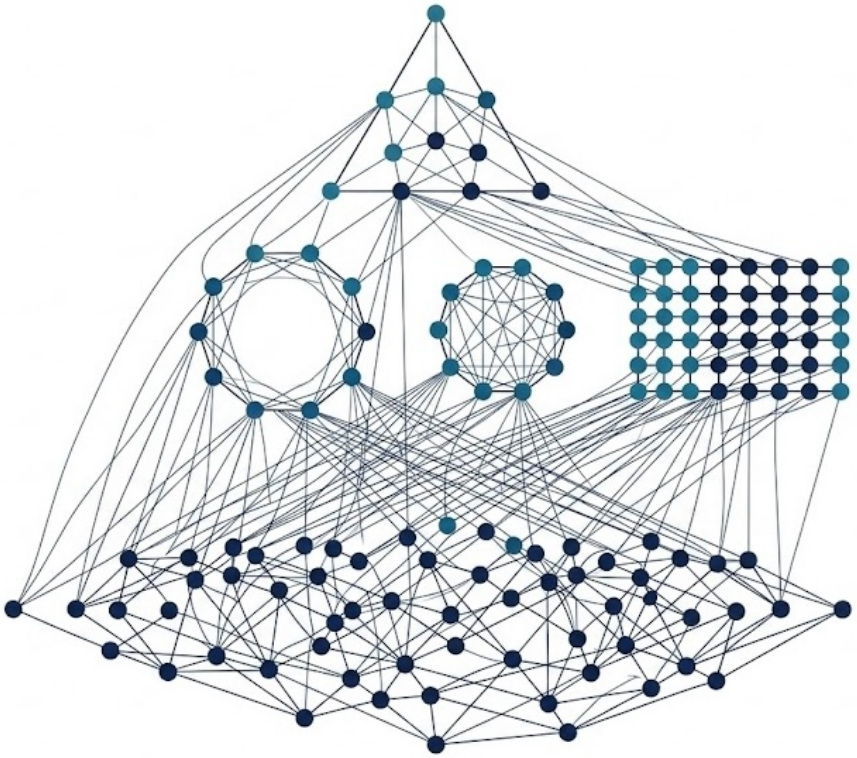

Strictly-Local Multi-Agent Formation Control via BC-Anchored Reinforcement Learning

Master's thesis in Computer science and engineering

Yihuai Cai

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's thesis 2026

Strictly-Local Multi-Agent Formation Control via BC-Anchored Reinforcement Learning

Yihuai Cai

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Strictly-Local Multi-Agent Formation Control via BC-Anchored Reinforcement Learning

Yihuai Cai

Supervisor: Giovanni Volpe, Physics Department

Examiner: Giovanni Volpe, Physics Department

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: From local interactions to global form — a decentralized robot swarm self-organizing into target shapes.

Typeset in LaTeX
Gothenburg, Sweden 2026

Strictly-Local Multi-Agent Formation Control via BC-Anchored Reinforcement Learning

Yihuai Cai
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Multi-robot formation control requires a group of robots to coordinate under limited communication and local sensing in order to form desired geometric structures. Classical artificial-potential or spring-based controllers are interpretable and stable, but are difficult to turn directly into trainable closed-loop neural policies. In contrast, learning formation behavior directly with multi-agent reinforcement learning often suffers from symmetry-induced local optima, unstable reward shaping, and poor exploration.

This thesis studies strict-local self-organization for multi-robot formation control, where each robot observes only local relative geometry, local edge-distance information, and its own role encoding, without access to global target vectors or centralized planning. We propose a controller-guided learning framework that combines DAgger behavior cloning with Multi-Agent Proximal Policy Optimization (MAPPO) refinement. A local spring controller is first constructed from the desired pairwise distances of the target template and used as the teacher policy. A graph neural network with recurrent memory is then trained on the closed-loop state distribution induced by the learned policy. Finally, MAPPO refines the policy using a formation-stress reward aligned with the teacher controller, while a frozen behavior-cloning policy serves as an MSE anchor to prevent destructive policy drift. The target shape is conveyed entirely through the desired pairwise distance on local communication edges, so a single policy can switch between formations without any global goal vector.

Experiments demonstrate that the proposed method achieves stable formation on 10-robot multi-shape tasks, reaching succ@$0.4d = 0.998$ and succ@$0.25d = 0.987$ across triangular, hexagonal, circular, and rectangular-grid target formations. Using a single model per team size, trained on all of its target formations, the method retains partial scaling on the non-circular formations: succ@$0.4d = 0.965$ and succ@$0.25d = 0.617$ for the 21-robot system, and succ@$0.4d = 0.525$ and succ@$0.25d = 0.254$ for the 28-robot system. Large circular formations, however, remain unsolved at these larger team sizes, suggesting an empirical feasibility boundary of fixed-local formation control: the approach handles shapes that stay locally observable at the chosen communication scale, but struggles when the target geometry becomes globally sparse relative to the communication radius.

Keywords: Multi-Robot Formation Control; Multi-Agent Reinforcement Learning; Behavior Cloning; Graph Neural Network.

Acknowledgements

I would like to express my sincere gratitude to Giovanni Volpe for his support, and especially to John Tember for his continuous guidance throughout my master's thesis. His regular supervision, insightful discussions, and valuable feedback during our meetings have been instrumental in shaping this work.

Yihuai Cai, Gothenburg, 2026-06-12

Contents

List of Figures

List of Tables

1

Introduction

Multi-robot systems are an important research direction in robotics and artificial intelligence. Their central idea is to coordinate multiple relatively simple robots to accomplish tasks that are difficult, inefficient, or costly for a single robot to complete alone. With the decreasing cost of embedded computing platforms, sensors, and wireless communication modules, the hardware barrier for building small-scale robot swarms has been substantially reduced. For example, low-cost platforms based on Raspberry Pi, Pi Pico, ESP32, and similar modules can already support basic sensing, local communication, and distributed control experiments. This trend makes multi-robot systems increasingly practical beyond expensive specialized platforms and creates a realistic foundation for low-cost and large-scale deployment. In applications such as search and rescue, cooperative mapping, environmental monitoring, intelligent warehousing, area coverage, and distributed perception, a single robot is often limited by sensing range, payload, battery life, and fault tolerance. A group of robots can compensate for these limitations through parallel execution, spatial distribution, and redundancy. Reviews on swarm robotics have therefore emphasized decentralization, robustness, scalability, and low-cost deployment as key advantages of multi-robot systems over single-robot systems [1].

1.1 Formation Control as the Core Problem

Formation control is one of the most classical and fundamental problems in multi-robot coordination [2]. It aims to make a group of robots maintain or form a desired spatial geometry during motion, such as a triangle, circle, rectangle, or line formation. Many higher-level cooperative tasks can be viewed as extensions built on top of formation control. Search and rescue requires robots to cover unknown areas with an appropriate spatial layout; cooperative mapping requires robots to maintain useful sensing baselines; patrol and monitoring tasks often require boundary or ring-like distributions; and warehouse transportation requires robots to maintain safe relative spacing in constrained spaces. Therefore, the ability to form stable, controllable, and switchable formations is a prerequisite for many multi-robot tasks.

Classical multi-agent research provides the first foundation for this problem. Reynolds showed that group-level flocking can emerge from simple local rules [3]. Olfati-Saber developed flocking and consensus methods that explain how neighbor interactions can produce coherent collective motion [4], [5]. Mesbahi and Egerstedt organized these

ideas using graph-theoretic tools, where the communication graph and its Laplacian structure determine what information can flow through a multi-agent team [6]. In formation control specifically, distance-based methods are especially relevant to this thesis because they describe the target shape through inter-agent distances rather than absolute target coordinates. Krick et al. studied stabilization of infinitesimally rigid formations, and Oh et al. surveyed position-, displacement-, and distance-based formation-control approaches [2], [7].

These works solved important theoretical parts of the formation problem: they provide interpretable local interaction laws, stability arguments, and a mathematical language for reasoning about graph topology and formation rigidity. However, they usually do not solve the exact setting considered here. A controller is often designed for one target structure, one interaction graph, or one set of carefully selected gains. Many methods also assume that the desired graph is known and feasible, or that agents can be assigned to fixed target positions. This is different from the goal of this thesis: a homogeneous robot team should switch between several shapes using one shared policy, without global target vectors, permanent identities, or a centralized planner.

1.2 Related Work and the Remaining Gap

A second line of work studies formation control in more realistic robotic settings. Alonso-Mora et al. considered distributed multi-robot formation control in dynamic environments, where robots maintain a formation while avoiding obstacles and adapting to changing conditions [8]. This work is relevant because it moves formation control closer to deployment, but its assumptions are still different from the problem studied here. The formation objective is specified as a geometric task to be tracked by a designed distributed controller, and the robots are not learning one shared policy that changes behavior only through local edge features. In contrast, this thesis treats formation shape as an input to a learned decentralized policy: the same actor must react to triangular, hexagonal, circular, or rectangular templates without receiving global target positions.

Learning-based swarm control reduces some dependence on hand-designed control laws. Hüttenrauch et al. studied deep reinforcement learning for swarm systems and showed that homogeneous agents can learn from local, permutation-invariant observations [9]. This assumption is close to the present thesis because both settings avoid fixed robot identities and use local information. However, their formulation is more general swarm behavior learning, while this thesis requires precise geometric formation matching under a fixed communication radius. For this task, reinforcement learning from scratch is not enough: many robots explore simultaneously, the reward is sensitive to small geometric errors, and symmetric agents can converge to locally reasonable but globally wrong arrangements. Therefore, this thesis does not rely on unguided reinforcement learning; it first uses a spring-controller teacher to create stable local behavior and then refines that behavior using a formation-stress reward.

Graph neural networks are a natural architecture for decentralized multi-robot

systems because each robot communicates over a graph. Gama et al. studied GNNs for decentralized controllers, showing that graph filters and message passing can represent distributed control policies [10]. Tolstaya et al. applied GNNs to learn decentralized controllers for robot swarms [11]. Li et al. used GNNs for decentralized multi-robot path planning, and Blumenkamp et al. emphasized that real-world multi-robot GNN policies must remain executable under limited communication and hardware constraints [12], [13]. These works justify the use of a graph-recurrent actor, but their main assumptions differ from this thesis in an important way: the graph network is usually used to solve navigation, path planning, or a fixed decentralized control problem. They do not specify how a robot should infer a switchable formation template when it has no global center, no global orientation, and no absolute target slot.

This difference is central. In many formation-control methods, the target is encoded through assigned positions, a desired formation graph, or a controller designed for one shape. In many learned GNN methods, the graph defines who communicates, but it does not by itself define the desired formation geometry. This thesis connects these two ideas by placing the desired pairwise distance directly on each local communication edge. The policy therefore receives shape information in the same local form used by a spring controller, while still being trained as a shared graph neural policy.

Imitation learning provides the third component. Behavior cloning can train a policy from an expert, but it suffers when the learned policy visits states that were not present in the original demonstrations. DAgger addresses this distribution-shift problem by collecting expert labels on states induced by the current learned policy [14], [15]. Here, DAgger is used with a specific assumption: the teacher is not a global planner and does not provide global targets, but a local spring controller built from the same pairwise distance errors used in the reward. Pure imitation would still be limited by this teacher, so the policy is later refined with MAPPO, while a frozen behavior-cloning policy anchors the update and reduces destructive drift [16], [17].

The resulting research gap can be stated sharply: existing work provides strong tools for distance-based formation control, swarm reinforcement learning, GNN-based decentralized policies, and imitation learning, but it does not provide a complete framework for learning a single homogeneous strict-local policy that can switch among multiple formation shapes using only edge-level desired distances, without global target vectors, permanent robot identities, or centralized planning.

1.3 Key Constraints in This Thesis

The research problem is therefore defined by four strict constraints. First, sensing and communication are local. Each robot can observe only nearby neighbors, and the communication radius is fixed to 180 mm. Second, the actor does not receive global target vectors, a global formation center, a global orientation, or absolute target positions. Third, the robots are homogeneous and share one policy, so the

method should not rely on manually assigned permanent identities. Fourth, the same learning framework should support multiple formation shapes, including triangular, hexagonal, circular, and rectangular-grid templates.

These constraints make the problem different from standard formation control and from generic multi-agent reinforcement learning. Classical distance-based controllers are interpretable but usually require task-specific design. Reinforcement learning is flexible but unstable when trained from scratch. Imitation learning provides a stable starting point but can copy the teacher's limitations. GNN policies fit local communication graphs but still need a task representation that tells the network which shape to form. The remaining gap is a practical pipeline that combines these components for strict-local, homogeneous, multi-shape formation learning under a fixed communication radius.

1.4 Research Motivation and Contributions

The motivation of this research is to build such a pipeline by connecting classical local control and learning-based policy optimization. A local spring controller is first constructed from the desired pairwise distances of the active formation template. It is used as an interpretable teacher, not as the final deployed controller. A graph-recurrent decentralized actor is then trained with DAgger behavior cloning on closed-loop states. Finally, the actor is refined with MAPPO using a formation-stress reward aligned with the same pairwise distance errors, while a frozen behavior-cloning policy acts as an anchor against unstable policy drift.

The main contribution of this project is the combination of shape representation, training strategy, and strict-local execution. First, the desired formation is represented through pairwise distances on communication edges, rather than through global target positions. Second, temporary slot assignment breaks symmetry among homogeneous robots without introducing permanent robot identities. Third, the spring-controller teacher and DAgger stage give the graph policy a stable behavioral prior. Fourth, MAPPO refinement optimizes the learned policy using a formation-stress reward that is consistent with the controller's local distance objective. Fifth, the evaluation tests whether one model per team size can handle multiple shapes under the same fixed communication radius, and it identifies where this locality assumption fails for large sparse circular formations.

In summary, this thesis does not claim to replace classical formation-control theory or to solve all swarm-learning problems. Its contribution is more specific: it studies how a homogeneous, decentralized robot team can learn multi-shape self-organization when the only task information available at execution time is local relative geometry, temporary role information, and edge-level desired distance. This directly addresses the gap between interpretable distance-based control and deployable learned GNN policies for low-cost robot swarms.

2

Theory

2.1 Problem Formulation and Notation

2.1.1 Multi-Robot System Dynamics

This thesis considers a team of N homogeneous robots moving in a two-dimensional plane. The position and heading of robot i at discrete time t are denoted by

$$p_i^t \in \mathbb{R}^2, \qquad \theta_i^t \in (-\pi, \pi). \tag{2.1}$$

The full team configuration is written as

$$p^t = (p_1^t, \ldots, p_N^t) \in \mathbb{R}^{N\times 2}, \qquad \theta^t = (\theta_1^t, \ldots, \theta_N^t). \tag{2.2}$$

Each robot follows a unicycle-like discrete-time model [18]. The continuous action of robot i is

$$a_i^t = (u_i^t, \omega_i^t) \in [-1, 1]^2. \tag{2.3}$$

Here, u_i^t is a normalized linear-speed command and ω_i^t is a normalized angular command.

Let $v_{\max}$ be the maximum forward speed, Δt the time step, and $\Delta\theta_{\max}$ the maximum rotation per step. The simulator first updates the heading and then moves the robot along the new heading:

$$\theta_i^{t+1} = \operatorname{wrap}\Big(\theta_i^t + \omega_i^t \Delta\theta_{\max}\Big), \tag{2.4}$$

$$p_i^{t+1} = p_i^t + u_i^t v_{\max} \Delta t \begin{bmatrix} \cos\theta_i^{t+1} \\ \sin\theta_i^{t+1} \end{bmatrix}. \tag{2.5}$$

In the experiments, the action is produced by a stochastic Gaussian policy during training and by its deterministic mean during evaluation. The policy output is passed through a tanh nonlinearity so that the executed action lies in $[-1, 1]^2$.

2.1.2 Local Observability and Communication Graph

The decentralized policy is executed under a strict locality assumption. At each time t, the robot team induces a communication graph [6],

$$G^t = (V, E^t_{\text{loc}}), \qquad V = \{1, \dots, N\}. \tag{2.6}$$

An edge exists when two robots are within the fixed communication radius:

$$E^t_{\text{loc}} = \{(i, j) : i \neq j, \|p^t_i - p^t_j\| < r_{\text{comm}}\}. \tag{2.7}$$

The communication radius r_{comm} is treated as a physical constraint rather than a tunable performance parameter. It is fixed to 180 mm.

Each robot receives a local observation rather than the global team state. For an edge $j \to i$, the edge feature describes robot j from the body frame of robot i. It contains normalized relative position, distance, bearing, relative heading, relative velocity, the source robot's latent code, and the desired pairwise distance for the assigned formation slots. The most important task-specific scalar is

$$d^{\text{des}}_{ij} = D[\rho_i, \rho_j], \tag{2.8}$$

where D is the pairwise distance matrix of the active formation template and ρ_i is the temporary slot assigned to robot i. Thus, the target shape is exposed through local desired-distance features rather than through a global target vector, formation center, or formation orientation.

2.1.3 Formation Templates and Pairwise Distance Specification

A target formation is represented by a template

$$T = (q_1, \dots, q_N) \in \mathbb{R}^{N\times 2}, \tag{2.9}$$

where q_k denotes the canonical position of slot k in the target formation. The template is used only through its pairwise distance matrix

$$D_{kl} = \|q_k - q_l\| \qquad D \in \mathbb{R}^{N\times N}. \tag{2.10}$$

This representation is invariant to global translation and rotation of the template. It is also convenient for decentralized control because a robot does not need to know the absolute target position of its slot; it only needs the desired distances between its own slot and the slots of its local neighbors.

For multi-shape training, the set of possible templates is denoted by

$$\mathcal{S} = \{T^{(1)}, T^{(2)}, \dots, T^{(K)}\}. \tag{2.11}$$

At the beginning of each episode, one template $T^{(k)}$ is sampled and its corresponding pairwise distance matrix $D^{(k)}$ is used to construct edge features and the formation stress reward.

2.2 Formation Stress Potential

Distance-based formation control is a classical way to encode a target [2], [7]. Given the local communication graph E^t_{loc}, the temporary slot assignment ρ, and the desired distance matrix D, the local formation stress potential is defined as

$$V(p^t) = \sum_{(i,j)\in E^t_{\text{loc}}} \left(\frac{\|p^t_i - p^t_j\| - D_{\rho_i\rho_j}}{d} \right)^2, \tag{2.12}$$

where d is the nominal target spacing used for normalization. The implementation uses a per-agent normalized version of this potential and returns its negative value as the reward. For robot i, the local stress is

$$V_i(p^t) = \frac{1}{Z_i} \sum_{j\in\mathcal{N}_i} \left(\frac{\|p^t_i - p^t_j\| - D_{\rho_i\rho_j}}{d} \right)^2. \tag{2.13}$$

Here, $\mathcal{N}^t_i$ is the set of neighbors of robot i and Z_i is a normalization factor. The final experiments use $Z_i = N - 1$, which keeps the reward scale stable across training settings. The team-level reward component is the negative mean stress:

$$r^t_{\text{stress}} = -\frac{1}{N} \sum_{i=1}^{N} V_i(p^t). \tag{2.14}$$

This reward is strictly local in the sense that each per-agent contribution depends only on the robot's slot, its local neighbors, and the desired pairwise distance to those neighbors.

2.2.1 Gradient-Flow Interpretation

The stress potential provides a direct connection between the learned reward and a classical spring controller. For a single pair (i, j), let

$$r_{ij} = \|p_i - p_j\|, \qquad e_{ij} = r_{ij} - D_{\rho_i\rho_j}. \tag{2.15}$$

Ignoring constant normalization factors, the gradient of e^2_{ij} with respect to p_i is

$$\nabla_{p_i} e^2_{ij} = 2e_{ij} \frac{p_i - p_j}{\|p_i - p_j\|}. \tag{2.16}$$

Therefore the negative gradient direction is

$$-\nabla_{p_i} V \propto \sum_{j\in\mathcal{N}_i} e_{ij} \frac{p_j - p_i}{\|p_j - p_i\|}. \tag{2.17}$$

This is exactly the direction of a distance-based spring force, up to a constant gain. When two robots are farther than desired, $e_{ij} > 0$, and the force pulls them together. When they are too close, $e_{ij} < 0$, and the force pushes them apart. The hand-coded teacher controller used in this research implements this local spring-force rule and converts the resulting force vector into a unicycle action.

2.2.2 Geometric Interpretation

The formation stress potential can be interpreted as a local geometric consistency measure. It does not require matching the current robot positions to an absolute target map. Instead, it asks whether the distances between locally connected robots agree with the distances specified by the active template. If all relevant local distances match the template distances, the local graph has low stress. If robots are arranged as a visually incorrect blob, a stretched line, or a collapsed cluster, many local pairwise distances disagree with the template and stress increases.

This makes the stress reward better aligned with the controller than generic density or shape-profile rewards. The teacher controller attempts to move each robot in the direction that decreases the same distance errors that appear in the reward. This alignment is one of the main reasons why reinforcement learning can refine the behavior-cloned policy instead of destroying it.

2.2.3 Graph Neural Networks for Decentralized Policies

2.2.3.1 Message Passing on Robot Communication Graphs

The decentralized policy is implemented as a graph neural network over the local communication graph [11]. Let x_i denote the node feature of robot i, and let e_{ji} denote the edge feature for the directed edge $j \to i$. A general message-passing layer can be written as [19]:

$$m_{ji}^{(\ell)} = \phi^{(\ell)}\left(h_j^{(\ell)}, e_{ji}\right), \tag{2.18}$$

$$h_i^{(\ell+1)} = \sigma\left(W^{(\ell)} h_i^{(\ell)} + \mathrm{AGG}\{m_{ji}^{(\ell)} : j \in \mathcal{N}_i\}\right), \tag{2.19}$$

where $h_i^{(\ell)}$ is the hidden feature of node i at layer (ℓ), ϕ is a message function, and AGG is a permutation-invariant aggregation operator such as sum, mean, or attention-weighted sum. Because the same message and update functions are shared across robots, the learned policy can be applied to different robot indices without changing parameters.

In this thesis, the main actor uses a Graph Attention Network followed by a GRU [20]. The node encoder first maps each robot observation into a hidden feature. Two graph attention layers then exchange local information over the communication graph using the edge attributes. A node-wise recurrent unit maintains temporal memory, and a node-wise action head outputs the continuous action mean for each robot.

2.2.3.2 Graph Attention and Edge-Aware Aggregation

Graph attention networks compute a weighted aggregation of neighbor messages, where the weights are learned from node and edge features [21]. For a target node i, the message from neighbor j is weighted by an attention coefficient

$$\alpha_{ji} = \frac{\exp(a(h_j, h_i, e_{ji}))}{\sum_{k \in \mathcal{N}_i} \exp(a(h_k, h_i, e_{ki}))}. \tag{2.20}$$

Here, $a(\cdot)$ is a learned scoring function. The next node feature is then formed from a weighted sum of neighbor messages. This allows the policy to treat different neighbors differently depending on their relative geometry, velocity, latent code, and desired slot distance.

2.2.3.3 Permutation Equivariance of the GNN Policy

A key property of shared-parameter graph policies is permutation equivariance. Let P be a permutation matrix that relabels the robot indices. For node features X, adjacency matrix A, and edge features E, a graph policy π_θ is permutation equivariant if

$$\pi_\theta(PX, PAP^\top, P \cdot E) = P\,\pi_\theta(X, A, E). \tag{2.21}$$

This means that relabeling the robots relabels the output actions in the same way, but does not change the physical behavior of the team. Message-passing GNNs satisfy this property when their message functions are shared across nodes and their aggregation functions are permutation invariant [22]. The recurrent and output layers in this thesis are applied node-wise with shared parameters, so they preserve the same equivariance property.

This property is important for homogeneous robot teams. The policy should not depend on arbitrary robot indices. Instead, it should depend on local geometry, local communication structure, and the temporary role encoding used for the current episode.

2.3 Reinforcement Learning

2.3.1 Markov Decision Processes and Policy Gradients

Reinforcement learning models sequential decision making as a Markov decision process [23]. At time t, an agent observes a state or observation, samples an action from a policy π_θ, receives a reward, and transitions to the next state.

The objective is to maximize the expected discounted return

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \gamma^t r_t\right], \tag{2.22}$$

where $\gamma \in [0, 1]$ is the discount factor. Policy-gradient methods optimize this objective by estimating gradients of the form [24]

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|o_t)\hat{A}_t\right], \tag{2.23}$$

where $\hat{A}_t$ is an advantage estimate. In a multi-robot setting, the joint reward depends on the behavior of the whole team, but the decentralized actor still produces one action per robot from local observations.

2.3.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO) stabilizes policy-gradient updates by limiting how far the new policy can move away from the data-collecting policy [16]. Let

$$r_t(\theta) = \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{\text{old}}}(a_t|o_t)}. \tag{2.24}$$

This is the likelihood ratio between the new and old policies. The clipped PPO objective is

$$\mathcal{L}^{\text{clip}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]. \tag{2.25}$$

The clipping term prevents excessively large policy updates, which is especially useful when training continuous actions for a multi-agent system. In this thesis, PPO is used after behavior cloning has already placed the actor in a reasonable behavior region.

2.3.3 Generalized Advantage Estimation

Generalized Advantage Estimation (GAE) is used to trade off bias and variance in advantage estimation [25]. Given the temporal-difference residual

$$\delta_t = r_t + \gamma V_\psi(s_{t+1}) - V_\psi(s_t), \tag{2.26}$$

GAE computes

$$\hat{A}_t = \sum_{\ell=0}^{T-t-1} (\gamma\lambda)^\ell \delta_{t+\ell}, \tag{2.27}$$

where $\lambda \in [0, 1]$ controls the bias-variance tradeoff. In the implementation, a centralized critic is used to estimate the value function, while the actor remains decentralized.

2.3.4 Centralized Training and Decentralized Execution

Centralized training with decentralized execution (CTDE) is a common paradigm in multi-agent reinforcement learning [26]. During training, the critic may use richer information about the full team state in order to reduce variance and improve credit assignment. During execution, however, each agent's actor must depend only on its own local observation. This separation is well suited to robotics: simulation or offline training can use additional information, but real robots should execute decentralized policies.

The actor in this thesis uses only local node and edge features. The critic, by contrast, aggregates global team-level features during training. This follows the CTDE principle and keeps deployment compatible with local observation constraints.

2.3.5 Multi-Agent PPO

MAPPO extends PPO to cooperative multi-agent settings with shared rewards, shared or decentralized actors, and centralized critics [17]. It has been shown to be a strong and stable baseline for cooperative multi-agent tasks. In this thesis, MAPPO is used as the refinement stage after DAgger behavior cloning. The policy is initialized from the BC actor, optimized with the clipped PPO objective, and regularized by a frozen BC anchor. This design is important because training PPO from scratch on the formation task is difficult: the initial policy rarely discovers stable formations under sparse or misleading reward signals.

2.4 Imitation Learning and BC-Anchored Reinforcement Learning

2.4.1 Behavior Cloning and Covariate Shift

Behavior cloning trains a policy to imitate expert actions by supervised learning [27]. Given expert-labeled data $\mathcal{D} = \{(o_t, a_t^\star)\}$, the standard continuous-action BC objective is

$$\mathcal{L}_{\mathrm{BC}}(\theta) = \mathbb{E}_{(o,a^\star)\sim\mathcal{D}} \left[\|\mu_\theta(o) - a^\star\|^2 \right], \tag{2.28}$$

where $\mu_\theta(o)$ is the policy mean. BC is simple and sample efficient, but it suffers from covariate shift. If the learned policy makes small mistakes, it may visit states that are not present in the expert dataset. The errors can then compound during closed-loop execution.

2.4.2 DAgger

DAgger addresses this covariate shift by collecting data on the state distribution induced by the learned policy [14], [15]. At each iteration, the current policy is executed, the expert labels the visited states, and the new data are aggregated into the training set. In this thesis, DAgger is implemented in a streaming form. The environment steps with the actor's own action, or with a beta-mixed action

$$a^{\mathrm{exec}} = \beta a^\star + (1-\beta)a^\pi, \tag{2.29}$$

where $a^\star$ is the local spring controller action and a^π is the actor action. The coefficient β is annealed from teacher-dominated execution to actor-dominated execution. The expert still labels the states visited during this process. This makes the BC policy more robust under closed-loop rollouts.

2.4.3 Anchored Policy Optimization

After BC, reinforcement learning can improve the policy with respect to the final task reward. However, unconstrained RL may drift away from the useful behavior learned from the teacher. This problem is particularly severe when the reward is not perfectly aligned with the visual formation quality. To reduce destructive drift, the

training objective includes an anchor term that keeps the current actor close to the frozen BC actor, in the spirit of trust-region policy optimization [28]:

$$\mathcal{L}_{\text{anchor}}(\theta) = \mathbb{E}\left[\| \tanh(\mu_\theta(o)) - \tanh(\mu_{\text{BC}}(o))\|^2\right]. \tag{2.30}$$

The full actor loss used in the MAPPO refinement stage is the PPO actor loss plus a weighted anchor penalty. For continuous Gaussian policies, this mean-squared action penalty can be viewed as a practical local surrogate for keeping the updated policy near the BC reference policy. It is not a strict replacement for a full KL constraint, but it is simple, stable, and directly matched to the continuous actions executed by the robots.

2.5 Theoretical Observations

2.5.1 Permutation Equivariance of Rank-Based Slot Assignment

The target template contains N slots, but the robots are homogeneous. The system therefore needs a way to provide temporary role information without relying on manually fixed robot identities. At the beginning of each episode, each robot receives a latent vector $z_i \in \mathbb{R}^4$. The slot assignment is computed from the rank of the first latent coordinate:

$$\rho_i = \text{rank}(z_i^{(1)}). \tag{2.31}$$

In the implementation, this is computed by a double argsort operation. Assuming no ties, each robot receives exactly one slot and every slot is assigned to exactly one robot.

2.5.1.0.1 Proposition 1. Let $\sigma \in S_N$ be any permutation of robot indices. Let $P_\sigma z$ denote the latent vectors after relabeling robots by σ. Then the rank-based slot assignment satisfies

$$\rho(P_\sigma z) = P_\sigma \rho(z). \tag{2.32}$$

2.5.1.0.2 Proof sketch. The rank of $z_i^{(1)}$ is the number of robots whose first latent coordinate is smaller than $z_i^{(1)}$. Permuting the robot indices changes the order in which the latent values are listed, but it does not change the set of values that are smaller than a given robot's value. Therefore each physical robot keeps the same rank under relabeling, and the vector of ranks is permuted in exactly the same way as the robots. This proves permutation equivariance of the slot assignment.

2.5.1.0.3 Corollary. The full decentralized policy is permutation equivariant under robot relabeling, provided that node features, edge features, and hidden states are permuted consistently. The rank-based slot assignment is equivariant, the edge desired distances are gathered from the permuted slot labels, the GNN message passing layers are permutation equivariant, and the GRU and action heads are applied node-wise with shared parameters. Thus relabeling the robots relabels the output actions but does not change the physical team behavior.

2.5.2 Local Alignment Between Spring Control and Stress Reward

The local spring controller and the formation-stress reward are built from the same pairwise distance errors. This creates a local alignment between the teacher action direction and the reward-improving direction. Let the spring controller produce a desired planar velocity direction

$$v_i^{\mathrm{ctrl}} = c \sum_{j \in \mathcal{N}_i} (\|p_i - p_j\| - D_{\rho_i \rho_j}) \frac{p_j - p_i}{\|p_j - p_i\|}, \tag{2.33}$$

where $c > 0$ absorbs spring gains and normalization constants. This direction is proportional to $-\nabla_{p_i} V$.

2.5.2.0.1 Proposition 2. In the small-step, heading-aligned limit, a policy velocity v_i^{π} that has positive inner product with the controller direction decreases the stress potential to first order:

$$\sum_i \langle v_i^{\mathrm{ctrl}}, v_i^{\pi} \rangle > 0 \quad \Longrightarrow \quad V(p^{t+1}) < V(p^t) + O(\Delta t^2). \tag{2.34}$$

2.5.2.0.2 Proof sketch. For a small position update $\Delta p_i = v_i^{\pi} \Delta t$, first-order Taylor expansion gives

$$V(p^{t+1}) = V(p^t) + \sum_i \nabla_{p_i} V(p^t)^{\top} \Delta p_i + O(\|\Delta p\|^2). \tag{2.35}$$

Since $v_i^{\mathrm{ctrl}} \propto -\nabla_{p_i} V$, the linear term is proportional to

$$-\Delta t \sum_i \langle v_i^{\mathrm{ctrl}}, v_i^{\pi} \rangle. \tag{2.36}$$

If the sum of inner products is positive, the first-order change in V is negative. For unicycle robots, the executed position update is constrained by the heading. The statement therefore assumes a heading-aligned regime in which the angular controller has already oriented the robot approximately toward the desired velocity direction. Outside this regime, the angular action can be interpreted as a transient alignment mechanism rather than a direct position update.

This proposition should not be read as claiming that the BC gradient and PPO gradient are exactly the same. The BC gradient can vanish near the teacher policy, while the PPO gradient depends on returns, advantage estimates, and the critic. The claim is more local and geometric: the teacher action direction is compatible with decreasing the same stress potential that the PPO reward maximizes.

2.5.3 Interpretation of the BC Anchor

The alignment above explains why the BC anchor is useful in this thesis. The anchor does not simply oppose reinforcement learning. Instead, it biases PPO updates toward a region of action space where the policy already produces motions that are

locally compatible with decreasing formation stress. Within this teacher-aligned basin, PPO can improve details of the behavior, reduce formation error, and adapt to the learned value function without drifting into unstable configurations.

This interpretation also explains why PPO from scratch is difficult. Before the policy has learned the teacher-like local geometry, random actions rarely produce useful formations, making the stress reward a challenging exploration signal. DAgger first moves the policy into a meaningful region of the policy space. The stress reward then supplies task-level improvement, while the BC anchor prevents the policy from moving too far away from the local spring behavior that makes the formation stable.

3

Methods

This chapter describes the learning framework used in this thesis. The goal is not only to train a neural controller, but to turn a simple local formation rule into a deployable decentralized policy. The main design principle is that the actor should execute from local information only: relative neighbor observations, temporary role information, and desired distances on communication edges. Global target vectors, global formation centers, global orientations, and centralized planner outputs are not provided to the actor.

The complete method has three stages, shown in Figure 3.1. First, a local spring controller is constructed from the desired pairwise distances of the active formation template. This controller gives stable and interpretable formation behavior, and is used as a teacher rather than as the final deployed policy. Second, a graph-recurrent actor is trained with DAgger behavior cloning on states visited by the current actor, so that the learned policy is exposed to its own closed-loop state distribution. Third, the behavior-cloned actor is refined with MAPPO using a formation-stress reward, while a frozen copy of the behavior-cloned policy acts as an anchor against unstable policy drift.

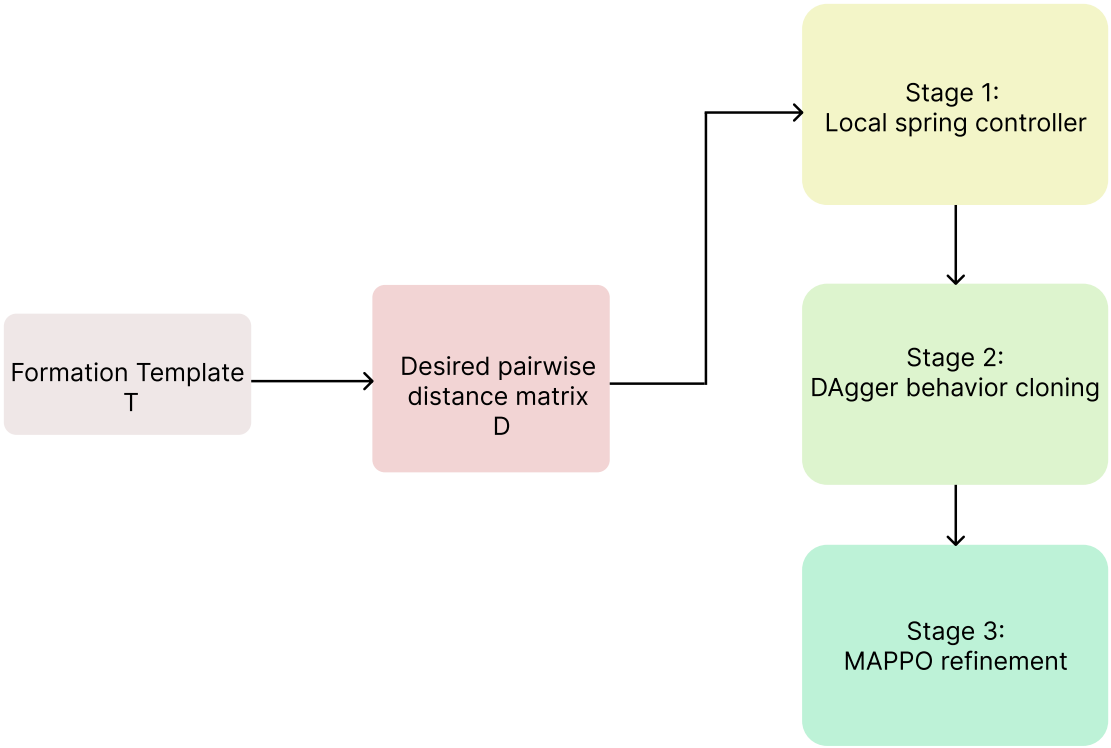

Figure 3.1: Training pipeline. A local spring controller provides closed-loop demonstrations, DAgger trains a graph-recurrent decentralized actor, and MAPPO refines the actor using the formation-stress reward with a frozen behavior-cloning anchor.

This structure is used because each stage solves a different difficulty in the formation problem. The spring controller provides a meaningful local geometric prior. DAgger reduces the distribution shift that would occur with ordinary offline imitation learning. MAPPO then improves the cloned policy with the actual task reward, while the anchor prevents the policy from moving too far away from the stable teacher-induced behavior.

3.1 Local Shape Representation

The target formation is represented by a template T, which is converted into a pairwise distance matrix $D(T)$. The policy does not receive the coordinates of the template slots. Instead, the active shape is communicated through the desired distance between temporarily assigned slots. For a directed communication edge $j \to i$, the task-specific edge feature is

$$d_{ji}^{\mathrm{des}} = \frac{D_{\rho_j \rho_i}}{d}, \tag{3.1}$$

where ρ_j and ρ_i are the temporary slots of the source and destination robots, and d is the nominal spacing used for normalization. Changing the formation therefore changes the distance matrix D, not the actor architecture.

This edge-level representation is the core connection between the classical controller and the learned policy. A distance-based spring controller acts on pairwise distance errors; the graph actor receives the same type of information as an input feature. This avoids giving the policy a global formation center or an assigned absolute target position, while still telling it which local geometric relationship should hold between two neighboring robots.

The node observation contains local motion and safety information together with a symmetry-breaking role code. Conceptually, the node feature for robot i is

$$o_i^{\text{node}} = [\text{local motion and safety features}, z_i, \text{onehot}(\rho_i)], \tag{3.2}$$

where $z_i \in \mathbb{R}^4$ is a per-episode latent vector and ρ_i is the temporary slot assignment. The local motion and safety features include body-frame velocity, nearest-neighbor distance, local collision-risk indicators, stuck-state information, and the previous action. These details are included for stable execution, but the important conceptual point is that all of them are local to the robot.

Because the robots are homogeneous, fixed manually assigned identities would be undesirable. The temporary slot assignment is therefore computed from the rank of the first latent coordinate,

$$\rho_i = \text{rank}(z_i^{(1)}). \tag{3.3}$$

This gives each robot a temporary role within the episode while keeping the policy shared across all robots. The slot is used in two places: as a node role encoding and as the index used to read the desired distance $D_{\rho_j \rho_i}$ on each edge. The assignment is fixed during one episode, but it is not a permanent robot identity. In a physical robot system, the same idea would require an initialization handshake or a distributed ranking protocol; the learned policy itself only requires the resulting temporary role code.

3.2 Local Spring Controller Teacher

The teacher is a hand-coded local spring controller. Its purpose is to provide stable formation behavior from the same pairwise distance information that the learned policy later observes. For each robot i, only neighbors inside the communication radius contribute to the control action. For neighbor j, define

$$r_{ij} = \|p_i - p_j\|, \qquad D_{ij}^{\rho} = D_{\rho_i \rho_j}. \tag{3.4}$$

The pairwise error is $r_{ij} - D_{ij}^{\rho}$, and the net controller force is

$$F_i = \sum_{j \in \mathcal{N}_i} k \left(r_{ij} - D_{ij}^{\rho} \right) \frac{p_j - p_i}{\|p_j - p_i\|}, \tag{3.5}$$

where k is the spring gain and $\mathcal{N}_i$ is the neighbor set of robot i. If two robots are farther apart than desired, the corresponding term pulls robot i toward robot j. If they are too close, the sign reverses and the term pushes robot i away. If the distance is correct, that pair contributes no force.

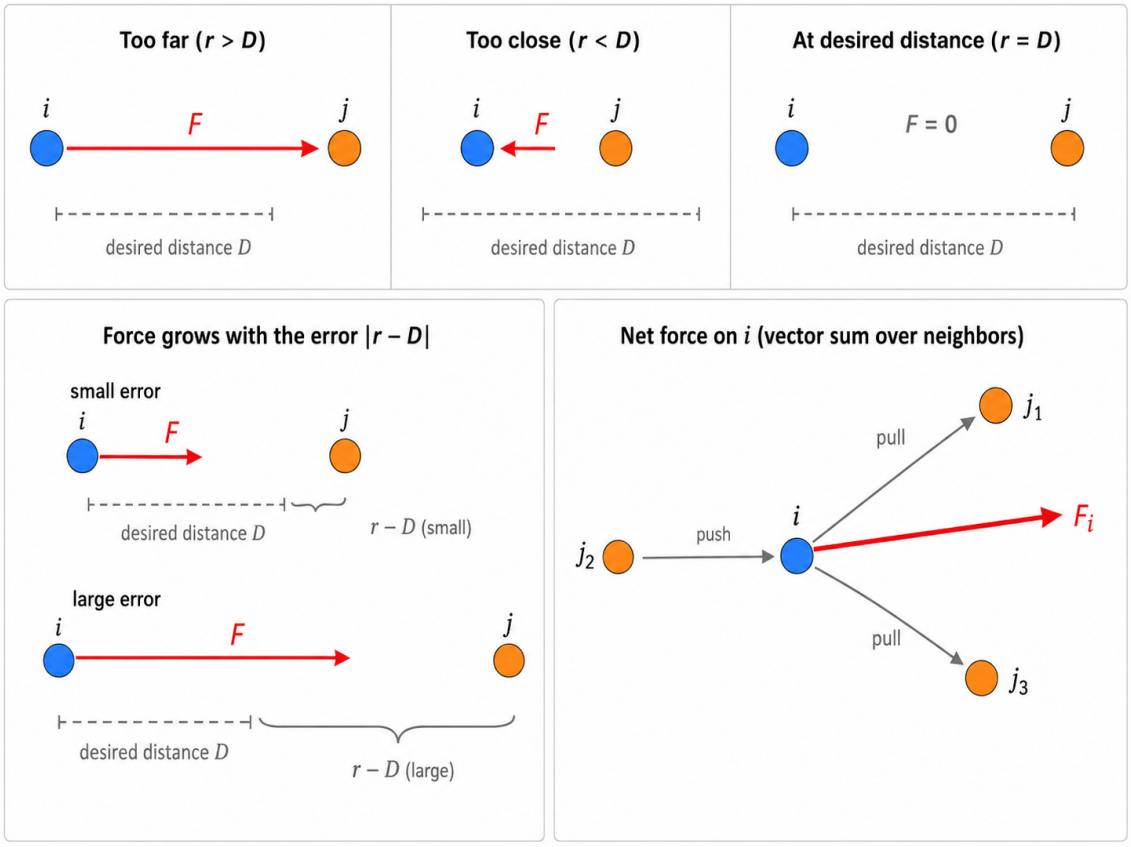

Figure 3.2: Behavior of the local spring controller. For a single neighbor, the force lies along the line between the two robots: when they are too far apart ($r > D$) it pulls robot i toward j, when too close ($r < D$) it pushes i away, and at the desired distance ($r = D$) it vanishes. With several neighbors, each contributes a component and robot i moves along their vector sum F_i.

The force is a control vector rather than a physical force. It is converted into the continuous unicycle action used by the robots. First, the desired heading is

$$\theta_i^{\text{des}} = \operatorname{atan2}(F_{i,y}, F_{i,x}). \tag{3.6}$$

The angular command turns the robot toward this heading:

$$\omega_i = \operatorname{clip}\left(\frac{\operatorname{wrap}(\theta_i^{\text{des}} - \theta_i)}{\Delta\theta_{\max}}, -1, 1\right). \tag{3.7}$$

The linear command is proportional to the force magnitude and is damped when the robot is not facing the desired direction:

$$u_i = \operatorname{clip}\left(\frac{\|F_i\|}{F_{\text{norm}}}, 0, 1\right) \max(0, \cos(\theta_i^{\text{des}} - \theta_i)). \tag{3.8}$$

When the force magnitude is small, the controller naturally commands a low speed. In the experiments, the spring gain is set to $k = 1.0$, and the force-to-speed scale used in the action conversion is $0.05d$. The resulting action is used as the supervision target during DAgger behavior cloning.

3.3 Graph-Recurrent Decentralized Actor

The learned actor is a graph-recurrent neural policy shared by all robots. This architecture is chosen because the robot team naturally forms a communication graph, and because the number and order of neighbors can vary over time. The actor receives node features, directed edge indices, edge attributes, and recurrent hidden states. It consists of a node encoder MLP, two edge-aware GAT graph convolution layers [21], a per-node GRUCell for temporal memory [20], and a velocity head that outputs the continuous action mean.

For robot i, the action mean can be written abstractly as

$$\mu_i = \mu_\theta(o_i, \{e_{ji} : j \in \mathcal{N}_i\}, h_i), \tag{3.9}$$

where o_i is the local node observation, e_{ji} are incoming edge features, and h_i is the recurrent hidden state. During training, the mean parameterizes a Gaussian policy with learned log standard deviation. During deterministic evaluation, the executed action is

$$a_i = \tanh(\mu_i). \tag{3.10}$$

The recurrent state is important because formation behavior is not purely reactive. Robots may need to remember recent motion, blocked behavior, or neighbor changes. The graph layers process the current local communication structure, while the recurrent state gives each robot short-term temporal context without introducing global information.

3.4 DAgger Behavior Cloning

The first learning stage trains the graph actor to imitate the local spring controller. Ordinary behavior cloning would collect expert trajectories once and train on those states. This is insufficient for closed-loop robot control, because the learned policy may later visit states that the expert demonstrations did not cover. DAgger addresses this problem by rolling out the current actor and asking the teacher to label the states that the actor actually visits [14], [15].

During data collection, the executed action can be a mixture of the controller action and the actor action:

$$a^{\text{exec}} = \beta a^{\text{ctrl}} + (1 - \beta) a^{\pi}. \tag{3.11}$$

The coefficient β is annealed from 1 toward 0 over the DAgger warmup period. Early collections are teacher-dominated, which keeps behavior stable. Later collections expose the policy to its own state distribution, which reduces the mismatch between training and evaluation.

The actor contains a GRU, so training uses rollout chunks with the hidden state recorded at the beginning of each chunk. This keeps the recurrent state during optimization consistent with the state used during execution. The behavior-cloning loss combines an action-matching term with a direction-matching term:

$$\mathcal{L}_{\text{BC}} = \mathbb{E}\left[w_t \| \tanh(\mu_\theta(o_t)) - a_t^{\text{ctrl}} \|^2\right] + \alpha_{\cos}\mathcal{L}_{\cos}. \tag{3.12}$$

The weight w_t down-weights low-speed teacher labels, and $\mathcal{L}_{\cos}$ penalizes disagreement between predicted and teacher action directions when the teacher speed is meaningful. The best behavior-cloning checkpoint is selected by deterministic multi-shape evaluation.

3.5 MAPPO Refinement and BC Anchor

The second learning stage initializes the actor from the best DAgger checkpoint and refines it with MAPPO. The actor remains decentralized and receives only the local node and edge features described above. A centralized critic is used during training to aggregate richer team-level information, following the centralized-training-decentralized-execution paradigm [26]. At evaluation time, only the decentralized actor is used.

The main reward signal is the formation-stress reward described in Chapter 2. The implementation combines an environment-level stress term, a per-agent stress term, safety penalties, and action penalties:

$$r_i^t = w_{\text{env}} r_{\text{stress,env}}^t + w_{\text{agent}} r_{\text{stress},i}^t + r_{\text{safety},i}^t + r_{\text{action},i}^t. \tag{3.13}$$

The per-agent stress reward is the negative normalized local distance error,

$$r_{\text{stress},i}^t = -\frac{1}{Z_i} \sum_{j \in \mathcal{N}_i^t} \left(\frac{\|p_i^t - p_j^t\| - D_{\rho_i \rho_j}}{d} \right)^2, \tag{3.14}$$

where $\mathcal{N}_i^t$ is the set of neighbors within the communication radius and Z_i is a normalization factor. The environment-level stress reward is the team average,

$$r_{\text{stress,env}}^t = \frac{1}{N} \sum_{i=1}^{N} r_{\text{stress},i}^t. \tag{3.15}$$

Safety is handled through collision and connectivity penalties,

$$r_{\text{safety},i}^t = -w_{\text{col}} c^t - w_{\text{conn}} q_{\text{iso}}^t, \tag{3.16}$$

where c^t is the collision-count diagnostic and q_{iso}^t is the number of isolated agents. The action term discourages excessive continuous commands,

$$r_{\text{action},i}^t = -w_{\text{lin}} (a_{i,\text{lin}}^t)^2 - w_{\text{ang}} (a_{i,\text{ang}}^t)^2. \tag{3.17}$$

In the final configuration, the weights are $w_{\text{env}} = 1.0$, $w_{\text{agent}} = 0.5$, $w_{\text{col}} = 0.2$, $w_{\text{conn}} = 0.5$, $w_{\text{lin}} = 0.10$, and $w_{\text{ang}} = 0.05$, with $Z_i = N - 1$. The reward uses stress as the primary shaping signal; legacy target and density rewards are not used by the strict-local actor.

A frozen copy of the behavior-cloned actor is used as an anchor during MAPPO refinement. For the same observation chunk, the current actor and the frozen behavior-cloning actor both produce continuous action means. The anchor loss is

$$\mathcal{L}_{\text{anchor}} = \mathbb{E}\left[\| \tanh(\mu_\theta(o)) - \tanh(\mu_{\text{BC}}(o)) \|^2 \right]. \tag{3.18}$$

The actor objective is therefore

$$\mathcal{L}_{\text{actor}} = \mathcal{L}_{\text{PPO}} + \lambda_{\text{BC}} \mathcal{L}_{\text{anchor}}. \tag{3.19}$$

In the main experiments, $\lambda_{\text{BC}} = 2.0$. This anchor does not replace the formation-stress reward. Instead, it acts as a stabilizer around the teacher-induced behavior, reducing the chance that PPO refinement destroys a visually correct formation policy while still allowing task-reward improvement.

3.6 Training and Evaluation Protocol

For each experiment setting, the training procedure is:

1. choose the formation set and team-size configuration;
2. construct template pairwise distance matrices;
3. train a DAgger behavior-cloning actor using the local spring controller as teacher;
4. initialize MAPPO from the selected behavior-cloning checkpoint;
5. load the same behavior-cloning checkpoint as the frozen anchor;
6. refine the policy with formation-stress reward and BC anchor;
7. select the best checkpoint using deterministic evaluation.

The 10-robot multi-shape experiments use triangular, hexagonal, circular, and rectangular-grid formations. The fixed-local scaling experiments use compact triangular, hexagonal, and rectangular-grid formations for $N = 21$ and $N = 28$. The communication radius is kept fixed at 0.180 meters for all scaling experiments.

Evaluation uses deterministic policy means rather than stochastic sampling. For each shape, the policy is evaluated over multiple random seeds and multiple parallel environments. The main metrics are

$$\text{succ@0.4d}, \quad \text{succ@0.25d}, \quad \text{match}(d), \quad \text{agent match}, \quad \text{collisions}/N.$$

The success metrics are computed from the final shape-matching distance after aligning the template by rotation and solving the assignment with Hungarian matching [29]. The loose threshold succ@0.4d measures approximate formation, while the strict threshold succ@0.25d better reflects visually precise formation. The final reported aggregate is the mean across per-shape averages.

4

Results

4.1 Experimental Setup

This chapter evaluates the final decentralized formation policy developed in this thesis. The same training pipeline is used throughout the main experiments: behavior cloning from the local spring controller, followed by MAPPO fine-tuning with the formation-stress reward and a frozen behavior cloning anchor. The actor is executed in a fully decentralized manner, and the communication radius is fixed at $r_{\text{comm}} = 180\,\text{mm}$ for all robot counts.

The evaluation uses deterministic policy execution. At the end of each episode, the final robot positions are centered at the swarm centroid. The target template is then rotated over a discrete grid of angles, without scaling or reflection, and the best robot-template assignment is computed by Hungarian matching. The following metrics are reported throughout this chapter.

Match d is the mean distance between each robot and its assigned template point, normalized by the nominal inter-robot spacing $d = 45\,\text{mm}$. It is the primary continuous measure of geometric accuracy. succ@$0.4d$ and succ@$0.25d$ are the fractions of trials whose matching distance is below $0.4d$ and $0.25d$, respectively. The former measures approximate formation completion, while the latter is used as the main strict success criterion. *Agent match* is the fraction of individual robots whose matched distance to a template point is below $0.25d$, capturing whether some robots are left behind even when the overall shape is recognizable. coll/N is the cumulative number of unordered robot-pair collision events during an episode, normalized by the number of robots. It should be read as an accumulated contact-frequency metric rather than as the number of robots that ended in collision: a short contact between one pair contributes one event, and repeated contacts over time increase the value.

Table 4.1: Evaluation protocol for the final method. The internal template names are omitted; the table uses descriptive formation names.

Robots	Formation set	Seeds	Parallel envs	Horizon
10	triangular, hexagonal, circular, rectangular grid	5	64	300
21	triangular, hexagonal, large circular, rectangular grid	5	64	600
28	triangular, hexagonal, large circular, rectangular grid	5	16	600

4.2 Ten-Robot Multi-Shape Formation

The 10-robot setting is the main multi-shape benchmark. It contains four qualitatively different formations: triangular, hexagonal, circular, and rectangular grid. Table 4.2 shows that the final policy achieves near-perfect approximate success on all four shapes. The strict success rate remains high as well, reaching 0.987 when averaged across the four formations.

Table 4.2: Per-formation results with 10 robots. Values are mean ± standard deviation over five random seeds.

Formation	succ@0.4d	succ@0.25d	Agent match	Match d	coll/N
Triangular	0.997 ± 0.007	0.997 ± 0.007	0.972 ± 0.005	0.092 ± 0.000	0.871 ± 0.012
Hexagonal	1.000 ± 0.000	0.997 ± 0.007	0.968 ± 0.008	0.095 ± 0.003	1.816 ± 0.208
Circular	0.997 ± 0.007	0.991 ± 0.009	0.990 ± 0.007	0.081 ± 0.004	0.861 ± 0.088
Rectangular	1.000 ± 0.000	0.963 ± 0.018	0.850 ± 0.013	0.146 ± 0.002	0.981 ± 0.115
Average	0.998	0.987	0.945	0.103	1.132

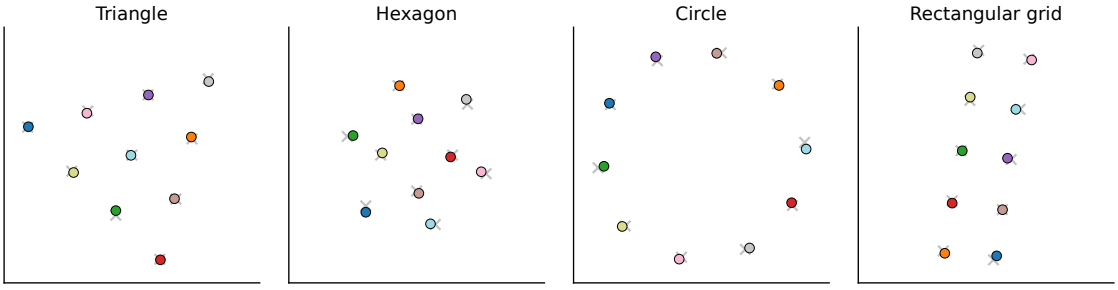

Figure 4.1: Final 10-robot formations produced by the learned decentralized policy. The gray crosses show the aligned target template and the colored circles show the final robot positions.

Figure 4.1 gives representative final configurations. The robots form all four shapes without global target assignments or centralized execution. The rectangular grid is the hardest of the four in strict matching, which is reflected by its larger average matching distance.

4.3 Scaling with Fixed Local Communication

The same method is next evaluated with larger teams while keeping the communication radius fixed at 180 mm. The purpose of this experiment is not to give the robots a larger communication graph as the team grows, but to test whether the same local learning formulation can still form larger formations when the number of robots increases. For each team size, a single model is trained on all target formations used for that team size and evaluated per-formation under the same fixed-local setting.

Table 4.3: Aggregate fixed-local scaling results for the final method. The large-team rows report the mean over the three non-circular formations (triangular, hexagonal, and rectangular grid); the large circular formation is reported separately in Table 4.4.

Robots	Aggregate set	succ@0.4d	succ@0.25d	Agent match	Match d	coll/N
10	triangular, hexagonal, circular, rectangular	0.998	0.987	0.945	0.103	1.132
21	triangular, hexagonal, rectangular	0.965	0.617	0.600	0.248	6.93
28	triangular, hexagonal, rectangular	0.525	0.254	0.351	0.517	4.69

Table 4.3 shows the main scaling trend. For the larger team sizes, the aggregate deliberately excludes the large circular formation because that shape behaves as a separate failure case and is analyzed in Table 4.4. This separation avoids hiding the partial scaling behavior of the non-circular formations, while still reporting the large circular results explicitly. Moving from 10 to 21 robots reduces strict success from 0.987 to 0.617 on the non-circular formations, while approximate success remains high at 0.965. At 28 robots, the task becomes substantially harder: succ@0.4d drops to 0.525, and strict success drops to 0.254. This indicates partial scalability under a fixed local communication constraint, rather than complete scale invariance. Figure 4.2 plots this trend.

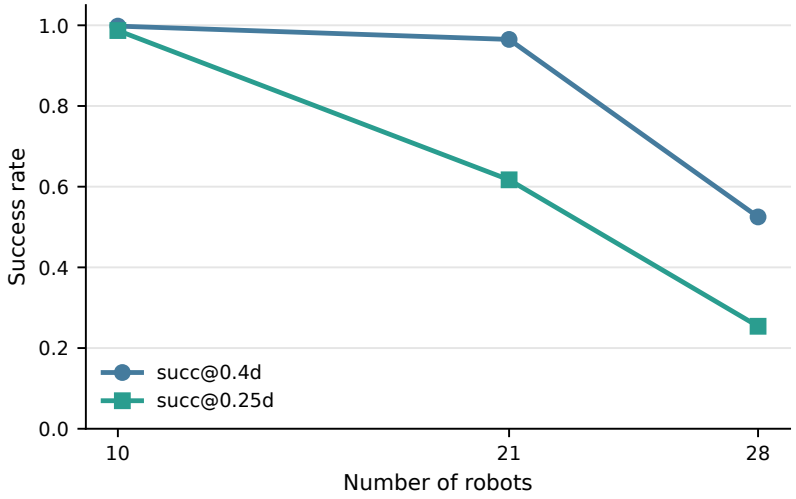

Figure 4.2: Success rate of the non-circular formations as the number of robots increases while the communication radius is held fixed. Both the approximate (succ@0.4d) and strict (succ@0.25d) success rates decrease with team size, with the strict rate dropping faster.

Table 4.4: Per-formation fixed-local scaling results at the 600-step evaluation horizon. For each team size, all rows come from one model trained on all listed formations. Values are means over five random seeds.

Robots	Formation	succ@0.4d	succ@0.25d	Agent match	Match d	coll/N
21	Triangular	0.975	0.822	0.674	0.225	6.60
21	Hexagonal	0.997	0.584	0.593	0.235	5.83
21	Rectangular grid	0.922	0.444	0.533	0.284	8.36
21	Large circular	0.037	0.000	0.071	1.628	4.25
28	Triangular	0.388	0.188	0.312	0.573	5.10
28	Hexagonal	0.688	0.225	0.364	0.428	4.39
28	Rectangular grid	0.500	0.350	0.378	0.550	4.58
28	Large circular	0.000	0.000	0.020	3.356	4.59

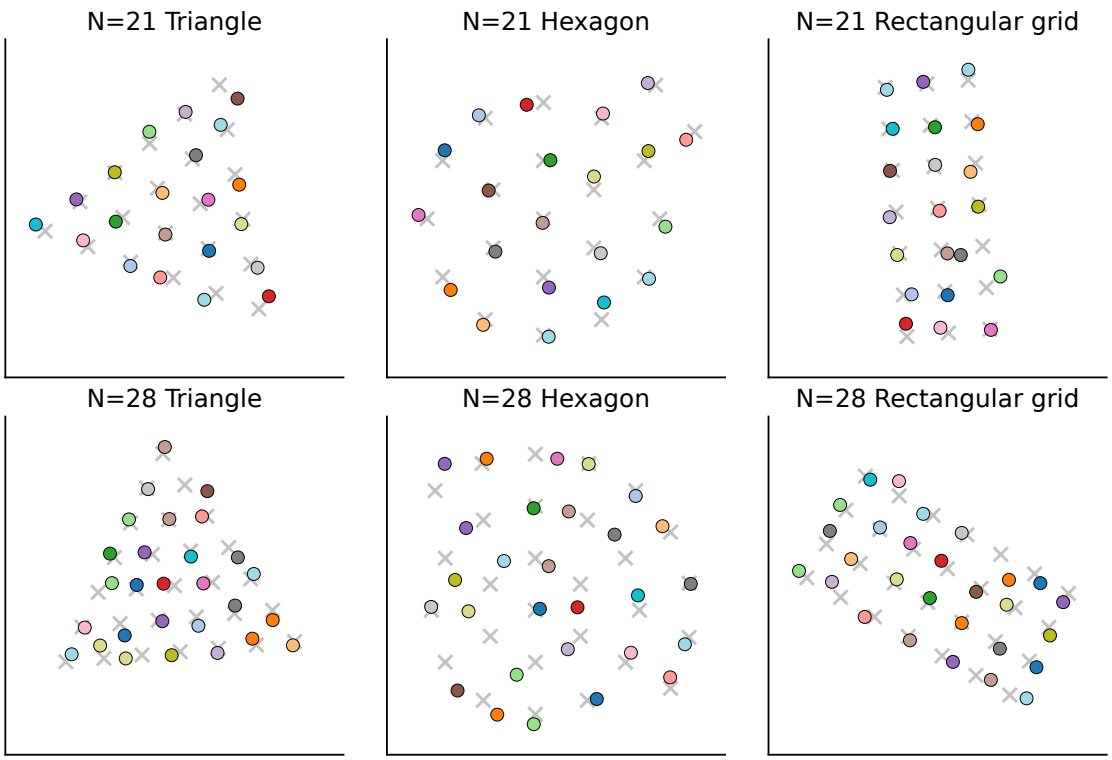

Figure 4.3: Representative final configurations of the non-circular formations for 21 and 28 robots, produced by the single model trained for each team size. The gray crosses show the aligned target template and the colored circles show the final robot positions. The larger teams still form recognizable structures, although the matching error and seed-to-seed variation increase with N.

The per-formation results in Table 4.4 show that the difficulty is not uniform across shapes. With 21 robots, the triangular and hexagonal formations retain high approximate success, while the rectangular grid remains recognizable but less precise. With 28 robots, all non-circular formations become harder. Figure 4.3 shows representative final configurations of these non-circular formations. The collision values in the larger-team rows are noticeably higher than in the 10-robot benchmark. Since coll/N accumulates pairwise contact events over the whole episode, these values indicate frequent transient contacts during formation, not necessarily final overlapping robots. The large circular formation fails at both scales, with zero strict success for both 21 and 28 robots.

4.4 Limitations on Large Circular Formations

Large circular formations are therefore the clearest limitation case in the large-team experiments. They are not evaluated with a separate model: the numbers in Table 4.4 come from the same N=21 and N=28 models used for the other large-team formations. The desired circular shape is globally sparse: many robots must occupy positions on a large loop while each robot still observes only a local neighborhood. The results

suggest that this geometry creates a stronger global coordination requirement than the triangular, hexagonal, and grid formations.

This limitation should be distinguished from the successful 10-robot circular formation reported above. With 10 robots, the circle remains a small formation relative to the fixed communication radius. With 21 or 28 robots, the circle has a much larger circumference while the communication radius is kept unchanged, turning the task into a sparse loop-like target. The observed failure therefore suggests an interaction between circular topology, team size, and fixed local communication, rather than an inability to represent circular templates in general.

The large circular rows in Table 4.4 show that this formation is not solved even when it is included in the training pool. The large circular formation obtains zero strict success at both 21 and 28 robots, whereas the non-circular formations remain partially learnable. The same models also produce high accumulated collision counts on the non-circular formations, which suggests that training with the difficult circular case may encourage more crowded transient motion before convergence. Figure 4.4 shows the corresponding final configurations: the robots do not close the target loop and instead form irregular clusters, with the mismatch becoming more severe at 28 robots.

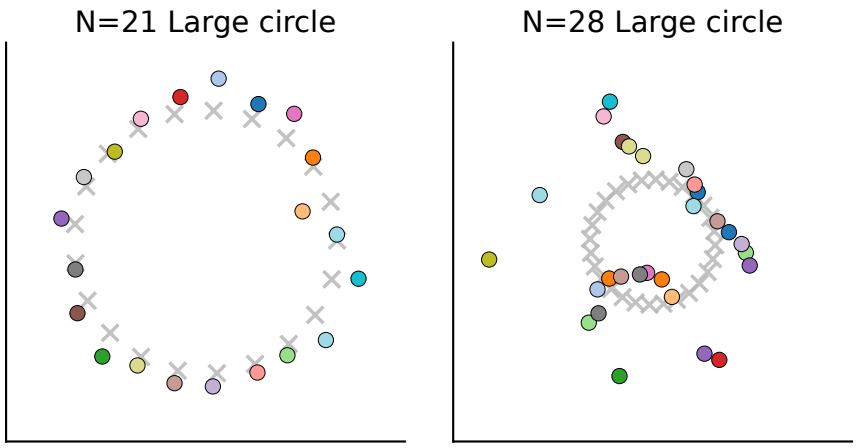

Figure 4.4: Final configurations for the large circular formation at 21 and 28 robots, produced by the same models used for the other formations. The robots do not close the loop: many positions on the target ring (gray crosses) stay empty while the robots form irregular clusters. The mismatch increases with team size as the ring circumference grows relative to the fixed communication radius.

5

Discussion and Conclusion

5.1 Discussion

5.1.1 Bridging Classical Local Control and Learned Decentralized Policies

The first observation from this thesis is that classical local control and learned decentralized policies should not be viewed as competing choices. Classical controllers are often interpretable and stable, but they are usually designed around a fixed task structure. Learning-based policies are more flexible, but they can be difficult to train from scratch in symmetric multi-robot systems. The method developed in this thesis combines these two sides: a local spring controller provides a stable behavioral prior, and a graph-recurrent policy absorbs this behavior through DAgger behavior cloning before being refined with MAPPO.

This design directly addresses the gap described in the introduction. Instead of discarding the classical controller, the learning system uses it in two ways. First, the controller supplies closed-loop demonstrations on states visited by the current policy. Second, the same pairwise distance errors used by the controller define the formation-stress reward for policy optimization. The frozen behavior-cloning anchor then keeps reinforcement learning within a teacher-aligned region of the action space. As a result, policy improvement is not an unconstrained search over unstable multi-robot behavior, but a refinement around a meaningful local-control basin.

The contribution is therefore not only the use of a graph neural network, but the training mechanism around it. The teacher gives the policy an initial understanding of local formation geometry; the stress reward gives a task-level objective; and the anchor prevents the refinement stage from drifting away from the stable local behavior. This suggests a practical route for applying learning to low-cost robot swarms: start from a simple local controller whose behavior is understandable, then use learning to make that behavior more flexible across shapes and team sizes.

5.1.2 Shape as a Local Edge Property

The second observation is that target shape information does not need to be given as a global goal vector. In this thesis, the actor does not receive a global formation center, global orientation, or assigned absolute target position. Instead, the shape is

represented locally through the desired pairwise distance d_{ij}^{des} attached to communication edges. This changes the role of the template: the template is not a map that each robot must navigate toward, but a source of local geometric constraints between temporarily assigned slots.

This design is important for combining strict locality with multi-shape formation. A single policy can be evaluated on different formations because the network architecture is unchanged; only the desired distance values on edges change. The same local message-passing mechanism can therefore process triangular, hexagonal, circular, and rectangular-grid templates. From the robot's perspective, the active formation is expressed through local questions: how far should I be from this neighbor, given our temporary slots? This makes the representation compatible with decentralized execution.

The edge-based representation also clarifies why rank-based slot assignment is useful. The private latent codes and their rank-induced slots break symmetry in a homogeneous team, while the pairwise distance matrix tells the policy what geometric relation should hold between two slots. The policy does not need a permanent robot identity, and it does not need global target coordinates. Shape "lives" on the edges of the local communication graph. This is the key reason why strict locality and multi-shape conditioning can coexist in the proposed framework.

5.1.3 A Feasibility Boundary for Fixed-Local Formation

The scaling experiments reveal a third observation: under a fixed local communication radius, formation performance depends not only on the number of robots or the category of the shape, but also on how the target geometry relates to the available communication scale. The method achieves reliable 10-robot multi-shape formation and retains partial scaling behavior at 21 and 28 robots on the non-circular formations. However, large circular formations fail at both larger team sizes, even though the 10-robot circular formation succeeds.

This contrast is best interpreted as evidence about the limits of the local representation, not as a proof of graph-theoretic impossibility. With 10 robots, the circular template remains compact relative to the communication radius. With 21 or 28 robots, the circumference becomes much larger while the communication radius remains 180 mm, producing a sparse loop-like target. The results suggest that, in this regime, the local graph may not provide enough coordination information for the learned policy to close the global circular structure reliably.

The broader lesson is therefore reflective rather than purely numerical. Edge-level desired distances are effective when the relevant geometric relationships remain visible through local communication. When a target shape requires long-range closure, as in the large circular formations, local pairwise constraints may no longer be sufficient by themselves. The experiments therefore suggest a feasibility boundary for fixed-local formation learning: strict locality can support multi-shape formation, but only when the shape remains sufficiently observable at the chosen communication scale.

5.1.4 Practical Considerations and Limitations

The final experiments follow a deliberately strict evaluation principle: for each larger team size, one model is trained on all target formations for that team size and evaluated on all of them. This avoids presenting separate models for easy and difficult shape sets. The cost of this choice is visible in the results. Including the large circular formation appears to make the learning problem harder, increases the accumulated pairwise contact events on the non-circular formations, and reduces strict geometric accuracy, especially at 28 robots. This is an honest cost of the single-model setting rather than a separate post-hoc limitation experiment.

Several limitations remain. First, all experiments are conducted in simulation. The communication radius is chosen to reflect a real small-robot constraint, but the experiments do not include real sensor noise, asynchronous wireless communication, actuation delays, wheel slip, or hardware failures. Second, evaluation uses deterministic policy means, which is appropriate for measuring the learned behavior but does not test robustness to stochastic execution noise. Third, collisions are measured as accumulated unordered pairwise contact events and penalized during learning, but safety is not enforced as a hard constraint. The high contact counts in the larger ring-inclusive models show that future work should treat safety more explicitly. Finally, the largest team size studied here is 28 robots, so larger-scale deployment remains an open question.

5.2 Conclusion

This thesis studied decentralized multi-robot formation learning under strict local observation, homogeneous robot identities, multi-shape target switching, and a fixed communication radius. The proposed method combines a local spring controller teacher, DAgger behavior cloning, an edge-aware graph-recurrent actor, a formation-stress reward, and a frozen behavior-cloning anchor for MAPPO refinement. The resulting policy uses only local node features, local edge features, and recurrent memory at execution time.

The work makes three main contributions. First, it shows how a classical local controller can be absorbed into a learned decentralized policy by using the controller both as a teacher and as the source of the reward structure. Second, it formulates shape conditioning as a local edge property: the active template is represented by desired pairwise distances on communication edges, rather than by global target vectors or absolute assigned positions. Third, it identifies an empirical feasibility boundary for fixed-local formation: locally dense formations remain learnable at larger team sizes, while large sparse circular formations appear to exceed what the current fixed-radius local representation can reliably coordinate.

Empirically, the method achieves strong 10-robot multi-shape formation, reaching $\text{succ@}0.4d = 0.998$ and $\text{succ@}0.25d = 0.987$ across triangular, hexagonal, circular, and rectangular-grid formations. With the same fixed communication radius, the larger-team models achieve partial scaling on the non-circular formations: $\text{succ@}0.4d = 0.965$

and succ@$0.25d = 0.617$ for 21 robots, and succ@$0.4d = 0.525$ and succ@$0.25d = 0.254$ for 28 robots. Large circular formations remain unsolved at both larger team sizes, with zero strict success, marking the main observed boundary of the current fixed-local approach.

Future work should focus on extending this boundary while preserving decentralized execution. The most direct directions are sparse long-range communication or hierarchical coordination for large loop-like formations, explicit safety layers or constrained reinforcement learning for collision avoidance, and real-robot experiments under sensing noise, communication delay, and actuation uncertainty. Larger teams, online formation switching, and dynamic obstacles would provide further tests of whether local edge-based shape conditioning can support practical robot swarm deployment.

Overall, the thesis shows that strict locality does not prevent learned multi-shape formation when the target geometry is represented in the right local form and the learning process is anchored to a stable local controller. At the same time, the large circular results show that locality has practical limits. Understanding and extending this boundary is the next step toward more capable decentralized robot swarms.

Bibliography

[1] 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, Mar. 2013. DOI: 10.1007/s11721-012-0075-2.

[2] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015. DOI: 10.1016/j.automatica.2014.10.022.

[3] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*, 1987, pp. 25–34. DOI: 10.1145/37401.37406.

[4] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006. DOI: 10.1109/TAC.2005.864190.

[5] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007. DOI: 10.1109/JPROC.2006.887293.

[6] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks* (Princeton Series in Applied Mathematics). Princeton University Press, 2010.

[7] L. Krick, M. E. Broucke, and B. A. Francis, "Stabilisation of infinitesimally rigid formations of multi-robot networks," *International Journal of Control*, vol. 82, no. 3, pp. 423–439, 2009.

[8] J. Alonso-Mora, E. Montijano, T. Nägeli, O. Hilliges, M. Schwager, and D. Rus, "Distributed multi-robot formation control in dynamic environments," *Autonomous Robots*, vol. 43, no. 5, pp. 1079–1100, 2019. DOI: 10.1007/s10514-018-9783-9.

[9] M. Hüttenrauch, A. Los, and G. Neumann, "Deep reinforcement learning for swarm systems," *Journal of Machine Learning Research*, vol. 20, no. 54, pp. 1–31, 2019.

[10] F. Gama, E. Tolstaya, and A. Ribeiro, "Graph neural networks for decentralized controllers," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5260–5264. DOI: 10.1109/ICASSP39728.2021.9414563.

[11] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," in *Conference on Robot Learning*, PMLR, 2020, pp. 671–682.

[12] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 11 785–11 792. DOI: 10.1109/IROS45743.2020.9341668.

[13] J. Blumenkamp, S. Morad, J. Gielis, Q. Li, and A. Prorok, "A framework for real-world multi-robot systems running decentralized gnn-based policies," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8772–8778. DOI: 10.1109/ICRA46639.2022.9811744.

[14] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.

[15] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[17] C. Yu et al., "The surprising effectiveness of ppo in cooperative multi-agent games," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 611–24 624, 2022.

[18] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*, 2nd ed. Springer, 2016.

[19] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International Conference on Machine Learning*, PMLR, 2017, pp. 1263–1272.

[20] K. Cho et al., "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[21] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[22] H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman, "Invariant and equivariant graph networks," in *International Conference on Learning Representations*, 2019.

[23] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[24] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, vol. 12, 2000, pp. 1057–1063.

[25] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[26] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[27] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Computation*, vol. 3, no. 1, pp. 88–97, 1991.

[28] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, PMLR, 2015, pp. 1889–1897.

[29] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

A

Appendix 1