Tampere University

Rong Guo

TOWARDS COLLABORATIVE OBJECT TRANSPORTATION USING DIFFERENTIAL DRIVE ROBOTS

A graph neural network based approach

Master's thesis
Faculty of Engineering and Natural Sciences
Associate Professor, Azwirman Gusrialdi
Professor, Alexandros Iosifidis
June 2026

ABSTRACT

Rong Guo
Master's thesis
Tampere University
Automation Engineering
June 2026

Collaborative object transportation requires groups of robots to carry objects to their goals while holding each object rigidly and avoiding collisions with other groups. The task becomes difficult as the number of groups grows: the groups need to coordinate in a shared space and keep formation while avoiding others in dense scenes.

A common approach uses an optimization framework, for example a Quadratic Program with Control Barrier Functions (QP-CBF), to enforce the formation and the safety distance as hard constraints. This gives safety guarantees for a small number of groups, but the hard constraints can conflict when many groups share a small space, leaving the solver with no feasible action. The groups then deadlock and fail to reach their goals. The per-step optimization also grows more expensive as the number of groups increases.

This thesis investigates whether a Graph Neural Network (GNN) can take over the task and scale to denser scenes. The system consists of groups of two differential-drive robots, where each group carries one object to its goal while avoiding the other groups. A GNN policy is trained using the QP-CBF controller as an expert, with physics-based loss terms, and is evaluated against the expert in single-group and multi-group scenarios.

In the single-group task, the policy reaches the goal in nearly all runs, the same as the expert, with a small loss of formation precision. In the multi-group task, the expert's success rate falls from 100% with two groups to 4% with eight, while the policy stays above 82% through ten groups, including group counts beyond its training range. The policy avoids the deadlock by turning away from other groups early and keeps working beyond its training range because each group only reacts to its nearby neighbors. It gives up some formation precision but keeps the groups moving and reaches the goals where the expert becomes infeasible.

Keywords: Collaborative object transportation, Graph Neural Network (GNN), unicycle, decentralized control, control barrier function (CBF), learning-based control

The originality of this thesis has been verified using the Turnitin Originality Check service.

USE OF AI IN THESIS

I have utilised AI tools in my thesis:

☒ No
☐ Yes

I acknowledge that I am fully responsible for the entire content of my thesis, including the parts generated by AI, and accept accountability for any violations of ethical standards in publications.

PREFACE

I would like to thank my supervisors, Professor Azwirman Gusrialdi and Professor Alexandros Iosifidis. Since last year, both of them have guided this work through our regular meetings every two weeks. I am grateful for their steady guidance throughout the project, for the time they gave whenever I ran into difficulties, and for working through the problems together with me. As my main supervisor, Professor Gusrialdi proposed the topic and provided the resources that made the training and experiments possible. Coming to this project without much background in machine learning, I am also grateful to Professor Iosifidis, whose guidance in this area helped me build the understanding needed to complete the work.

Finally, I would like to thank my family and friends for their support during my studies.

Tampere, 21 June 2026

Rong Guo

CONTENTS

1 INTRODUCTION 1
2 THEORETICAL BACKGROUND 3
2.1 Graph Neural Networks 3
2.2 Graph Attention Networks 4
2.3 Dataset Aggregation 5
3 PROBLEM FORMULATION 7
3.1 System Model 7
3.2 System Constraints 8
3.3 Assumptions 9
3.4 Problem Statement 10
4 PROPOSED METHOD 12
4.1 Graph Construction 13
4.1.1 Node Set 13
4.1.2 Edge Set 14
4.1.3 Node Features 15
4.2 Network Architecture 17
4.2.1 Node Encoder 18
4.2.2 GATs 19
4.2.3 Action Heads 20
4.2.4 Code Summary 21
4.3 Loss Function 22
4.3.1 Kinematic Predictor 22
4.3.2 Behavioral Cloning Loss 23
4.3.3 Navigation Losses 23
4.3.4 Formation Loss 25
4.3.5 Safety Loss 27
4.3.6 Composite Loss and Weight Allocation 29
4.4 Training Procedure 29
4.4.1 Data Composition 29
4.4.2 Dagger Aggregation Strategy 30
5 SIMULATION & DISCUSSION 31
5.1 Simulation Setup 31
5.1.1 Environment and Parameters 31
5.1.2 Baselines 32
5.1.3 Evaluation metrics 32
5.2 Scenario 1: Single-Group Collaborative Transport 34
5.2.1 Results 34
5.2.2 Analysis 39
5.3 Scenario 2: Multi-Group Collaborative Transport 40
5.3.1 Results 41
5.3.2 Analysis 49
5.4 Discussion 51

6 CONCLUSION....54
6.1 Limitations....54
6.2 Parameter Tuning....55
6.3 Future Work....56
REFERENCES....58

LIST OF SYMBOLS AND ABBREVIATIONS

BC	Behavioral Cloning
CAM	Control Admissibility Model
CSV	Comma-Separated Values
DAgger	Dataset aggregation
DR	Deadlock rate
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GNN	Graph Neural Network
MLP	Multilayer Perceptron
MSE	Mean Squared Error
QP-CBF	Quadratic Program with Control Barrier Function
ReLU	Rectified Linear Unit
SR	Success rate

$a^{(l,s)}$	Learnable attention vector (layer l, head s)
$\mathrm{b}_1, \mathrm{b}_2$	Encoder bias vectors
B	Set of robot samples in the training batch
c_m	Turning-circle center of the leader in group m
$c_{vel}, c_{lock}, c_{face}$	Scale factors in the loss terms
C	Set of valid inter-group leader pairs
d	Node feature dimension $(d = 10)$
d_h	Hidden embedding dimension $(d_h = 128)$
d_m	Inter-robot distance within group m
$\hat{d}_m$	Predicted leader-to-goal distance
$\hat{d}_m^{form}$	Predicted inter-robot (formation) distance
d_{min}	Minimum inter-group distance
d_{safe}	Minimum safe distance between turning centers
D_{form}	Formation-error scale (normalization constant)
D_{max}	Workspace diameter (normalization constant)
$e_{ij}^{(l,s)}$	Unnormalized attention score from node j to node i (layer l, head s)
$\bar{e}_{face}$	Mean face angle error
$\bar{e}_{form}$	Mean formation distance error
$E(t)$	Edge set of the graph
$E_{intra}, E_{goal}, E_{inter}$(t)	Intra-group, goal, and inter-group edge sets
g_m	Goal position of group m
$G(t)$	Graph at time t
$h_i^{(0)}$	Initial node embedding from the encoder
$h_j^{(l-1)}$	Neighbor embedding from the previous layer
$h_i^{(l)}$	Node embedding at GAT layer l
h_i^{out}	Final node embedding after the residual connection
k_v	Speed-decay constant in the speed profile
L	Total (composite) loss
L_v, L_ω	Behavioral cloning losses
$L_{goal}, L_{stop}, L_{head}, L_{form}, L_{arm}, L_{safety}$	Individual loss terms
L_{vel}, L_{lock}	Velocity-tracking and parking-lock components of the stop loss
L_{arm}^{base},L_{face}	Base arm and face components of the arm-and-face loss
L_o	Target formation distance (object length)
N	Number of groups

$obs_{m,i}^{x}$, $obs_{m,i}^{y}$	Displacement to the nearest robot of another group
$p_{m,i}$	Planar position of robot i in group m
$\hat{p}_{m,i}$	Predicted position of robot i in group m
$\hat{p}_{m,1}^{look}$	Predicted look-ahead position of the leader
P	Set of group indices in the training batch
r_{comm}	Communication and sensing radius
r_{min}	Minimum turning radius
r_{safe}	Body safety radius
s_m	Sign selecting the turning side of the leader in group m
$\mathbf{u}_{m,i}$	Control input of robot i in group m
S	Number of attention heads
T	Number of time steps in an episode
$\bar{T}$	Average navigation time
T_{look}	Look-ahead horizon in the safety loss
T_{pred}	Prediction horizon (Euler step size)
$u_{m,i}$	Control input of robot i in group m
$u_{m,i}^{*}$	Expert (reference) control action
$\tilde{u}_{m,i}$	Normalized (tanh-bounded) action
$v_{m,i}$	Linear velocity of robot i in group m
$v_{m,i}^{*}$	Expert linear velocity
$v_{m,i}^{tgt}$	Target linear velocity in the stop loss
v_{cmd}	Commanded speed profile
V	Node set of the graph
V_{max}	Maximum linear velocity
$viol_{mn}$	Barrier violation between groups m and n
W_1, W_2	Encoder weight matrices
$W^{(l,s)}$	Head-specific projection matrix (layer l, head s)
$\mathbf{x}_{m,i}$	Node feature vector of robot i in group m
$X(t)$	Node feature matrix
$\mathbf{z}_{m,i}$	Action-head input (robot and goal embeddings)
α_m	Relative heading angle between the two robots in group m
$\alpha_{ij}^{(l,s)}$	Normalized attention weight from node j to node i (layer l, head s)
β_m	Angle of the leader-to-goal vector at the predicted next position
Δt	Simulation time step
$\Delta\theta_m$	Wrapped difference between the two predicted headings in the face loss
$\Delta x_{m,i}^{g}$, $\Delta y_{m,i}^{g}$	Planar displacements from robot (m,i) to the goal
ϵ_{arm}	Angular tolerance in the arm loss
ϵ_{face}	Face-to-face heading tolerance
ϵ_{form}	Formation success tolerance
ϵ_{goal}	Goal-reaching tolerance
ϵ_{park}	Parking tolerance
$\theta_{m,i}$	Heading angle of robot i in group m
$\hat{\theta}_{m,i}$	Predicted heading of robot i in group m
$\hat{\theta}_{m,i}^{mid}$	Predicted midpoint heading in the Euler step
λ_v, λ_ω	Behavioral cloning loss weights
λ_{goal}, λ_{stop}, λ_{head}, λ_{form}, λ_{arm}, λ_{safety}	Loss weights
ρ_i	Role indicator (leader, follower, or goal)
σ	ReLU activation function

ψ_m	Angle from the follower's predicted position to the leader's predicted position in the arm loss
$\omega_{m,i}$	Angular velocity of robot i in group m
$\omega^*_{m,i}$	Expert angular velocity
Ω_{max}	Maximum angular velocity
i	Robot index within a group, $i \in \{1,2\}$
m	Group index
n	Index of a second (other) group
l	GAT layer index
s	Attention head index
k	Message-passing hop index
$\|\cdot\|$	Euclidean norm
$\|\cdot\|$	Absolute value / set cardinality
$\|$	Vector concatenation
$\odot$	Element-wise multiplication
$sg(\cdot)$	Stop-gradient operator
$Wrap\ (\cdot)$	Angle-wrapping function to $(-\pi, \pi]$
$sign(\cdot)$	Sign function
$1[\cdot]$	Indicator function
$1_{park}(m), 1^F_{park}(m)$	Leader and follower parking indicators
$N(i)$	Neighborhood of node i in the graph

1 INTRODUCTION

Multi-robot systems have been studied for a long time, because a team of robots has advantages over a single robot. They can be more flexible and more robust to the failure of one unit. They can also carry objects that are too large or too heavy for one robot alone. For this reason, collaborative object transportation has become an active research topic. Several robots move an object to a goal together. A common setting is a group of two robots that carry one object between them. The two robots must keep a relevant distance and heading angle apart, so that the object stays rigid while they move. When several such groups work in the same space, each group must also reach its own goal without colliding with the others.

One control method for this task is presented. It uses a quadratic program and control barrier functions (QP-CBFs) [1]. The formation distance between the two robots, the angle between them, and the safety distance to obstacles and to other groups are all written as hard constraints. At each step the controller solves the QP for an action that satisfies every constraint, which gives a mathematical guarantee of safety. The method is shown on groups of differential-drive robots in both simulation and real experiments. However, the examples in that work cover only up to three groups. The more groups' scenarios haven't been explored.

The QP-CBF method is one instance of optimization-based control. Model Predictive Control [2] and distributed optimization frameworks [1] have also been applied to multi-robot coordination tasks. These methods provide strict guarantees, but they all solve a constrained optimization problem at every step. The constraint set grows as the number of robots and inter-group couplings increases. In multi-group transport, formation maintenance, inter-group collision avoidance, and goal tracking must be handled at the same time. This computational cost makes real-time deployment hard.

Other works approach the transportation task in different ways. One studies a similar setting where two mobile manipulators carry a rigid rod together [3]. They use a leader-follower approach and solve the inverse kinematics for the end-effector positions to keep the load rigid, and they measure success by how well the distance between the two end-effectors stays equal to the length of the load. This setting is close to ours, but it controls manipulator arms not the mobile bases.

A different line of work does not address object transportation directly but shows how graph neural networks (GNNs) can be used for multi-robot control. It uses a GNN to solve large-scale motion planning, where many robots each move to a goal while avoiding collisions [4]. Because the network uses only local information, it can be trained with a small number of robots and then applied to a larger number without retraining. The robots in that work move independently and are not required to hold a formation, and they are modeled with simple dynamics that can move in any direction. Another one uses a GNN in a related way for decentralized formation control, where the network is trained with a fixed number of robots but still works for teams of different sizes [5].

These GNN works suggest a way forward for our problem. The QP-CBF method gives strong safety guarantees, but it was only shown for a few groups, and it can deadlock when the space becomes crowded. A GNN uses local information and has been shown to scale from a small number of robots to a larger one. This raises the question of whether a GNN could take over the same transportation task and handle the dense, many-group scenarios where the QP-CBF method struggles.

This thesis studies whether a GNN can take over the collaborative object transportation task. The system is a group of differential-drive robots carrying one object, where the object is represented by constraints on the distance between the robots and the angle from the object to each robot, and where each group must avoid other groups. We train a GNN and evaluate it in single-group and multi-group scenarios. The work is guided by three research questions:

- Can a GNN learn the single-group task of keeping the formation and reaching the goal, at a level comparable to the QP-CBF controller?
- Can a GNN avoid the deadlock that stops the QP-CBF controller in dense multi-group scenes, and still drive the groups to their goals?
- Does the learned policy still work when there are more groups than it saw during training?

2 THEORETICAL BACKGROUND

This chapter introduces the key techniques used to build and train the proposed controller. Section 2.1 describes GNNs and how they process graph-structured data through message passing. Section 2.2 presents Graph Attention Networks (GATs), which extend this framework with a learned attention mechanism. Section 2.3 introduces Dataset Aggregation (Dagger), which is used in the training process.

2.1 Graph Neural Networks

A graph $G = (V, E)$ consists of a set of nodes V and a set of edges E. Nodes represent entities in the system and edges represent the relationships between them. Each node i is associated with a state vector $\mathbf{x}_i \epsilon \mathbb{R}^d$ that contains the attributes of that entity. [6]

A Graph Neural Network (GNN) is a neural network that operates directly on graph-structured data.[6] Its encoding process follows three stages: a pre-processing step, iterative message passing, and post-processing layers. After that, a decoder produces the final output. [7]

In the pre-processing step, each node attribute x_i is mapped to a node feature vector $\tilde{\mathbf{x}}_i$ through a stack of neural networks. Then at GNN each layer, every node aggregates information from its neighbors and updates its own representation by message passing. For a k-layer GNN, the representation of node i at layer k is:

$$\mathbf{h}_i^{(k)} = \phi(\Lambda(\mathbf{h}_i^{(k-1)}, \oplus_{j \in \mathcal{N}_i} u_{ij})) \tag{2.1}$$

Where $\mathbf{h}_i^{(0)} = \tilde{\mathbf{x}}_i$, $\mathcal{N}_i$ is the set of neighbours of node i, $\oplus$ is a permutation-invariant aggregation function such as sum, mean, or max, u_{ij} is a feature vector describing the interaction between node i and node j, Λ determines how the node combines its own features with the aggregated neighborhood information, and ϕ is a learnable update function [7]. Figure 1 illustrates this process for a single node.

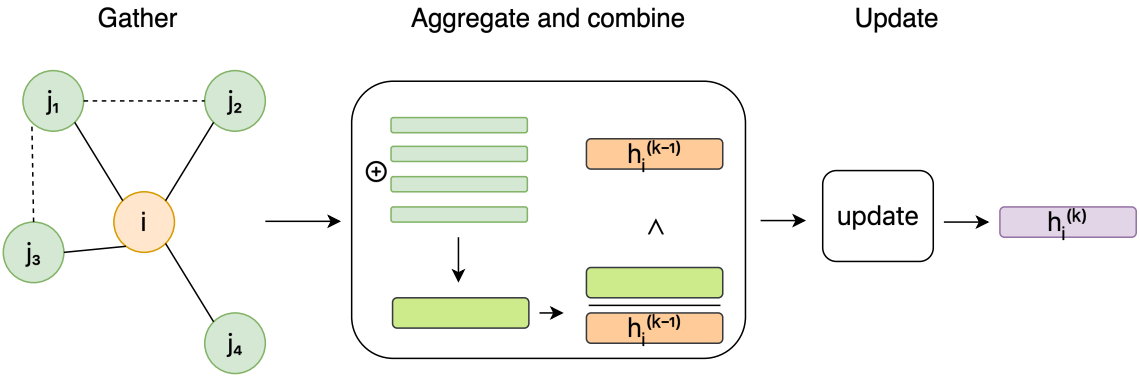


Figure 1. *Massage Passing.*

Figure 1 shows one message-passing step at node i, neighbors $j \in \mathcal{N}_i$ send their features u_{ij} to node i, these are aggregated by $\oplus$ and combined with node i's own representation $\mathbf{h}_i^{(k-1)}$ by $\wedge$. The result is passed through the update function ϕ to produce $\mathbf{h}_i^{(k)}$.

By stacking k such layers, each node's final layer's representation $\mathbf{h}_i^{(k)}$ captures information from nodes up to k hops away in the graph [8]. The post-processing layers then map the $\mathbf{h}_i^{(k)}$ into the node embedding $\mathbf{z}_i$ [7].

The final node embeddings $\mathbf{z}_i$ are passed to a decoder, which maps each $\mathbf{z}_i$ to a task-specific output. For classification tasks this is typically a softmax layer, while in our case for the regression tasks a nonlinear network is used to produce a continuous output [7].

2.2 Graph Attention Networks

In the standard GNNs described in Section 2.1, the aggregation function $\oplus$ treats all neighbors equally. However, in many tasks some neighbors carry more relevant information than others. GATs address this by introducing a learned attention mechanism into the aggregation step, allowing each node to assign different weights to different neighbors [9].

At each layer, the attention coefficient between node i and its neighbor j is computed as:

$$e_{ij} = \text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i] \parallel [\mathbf{W}\mathbf{h}_j]) \tag{2.2}$$

Where $\mathbf{W}$ is a shared weight matrix that transforms the node features, $\parallel$ denotes concatenation, and $\mathbf{a}$ is a learnable attention vector. The coefficients are then normalized across all neighbors of node i using softmax:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \tag{2.3}$$

The updated representation of node i is then computed as a weighted sum of the transformed neighbor features:

$$\mathbf{h}_i' = \sigma(\sum_{j \epsilon \mathcal{N}_i} \alpha_{ij} \mathbf{W} \mathbf{h}_j) \tag{2.4}$$

where σ is a nonlinear activation function [9].

To stabilize the learning process and increase the expressive capacity of the model, GAT uses multi-head attention. Each head m computes its own set of attention coefficients α_{ij}^m and transformed features $\mathbf{W}^m \mathbf{h}_j$ independently. The outputs from M heads are then concatenated:

$$\mathbf{h}_i' = \|_{m=1}^{M} \sigma\left(\sum_{j \epsilon \mathcal{N}_i} \alpha_{ij}^m \mathbf{W}^m \mathbf{h}_j\right) \tag{2.5}$$

Where $\|$ denotes concatenation across heads. At the final layer of the GAT, concatenation is typically replaced by averaging to produce a single output vector [9].

2.3 Dataset Aggregation

The traditional approach trains a policy $\hat{\pi}$ to perform well under the distribution of states visited by the expert π^* using supervised learning:

$$\hat{\pi} = \arg\max_{\pi \in \Pi} \mathbb{E}_{s \sim d_{\pi^*}}[\ell(s, \pi)] \tag{2.6}$$

where Π is the class of candidate policies, $\mathbb{E}$ denotes the expectation, d_{π^*} is the distribution of states encountered by the expert, and $\ell(s, \pi)$ is the loss of policy π in state s with respect to the expert's action [10].

However, this approach may have low performance guarantees. Because the policy is trained under the expert's state distribution d_{π^*} but executed under its own distribution d_{π}. Small prediction errors will cause the policy to drift into states the expert never visited, where it tends to make further errors, so the errors compound along the whole process. Because of this compounding, the policy's total cost relative to the expert can grow fast with the task horizon T [10].

DAgger addresses this by iteratively collecting new training data under the learned policy. At each round, the current policy is executed to generate trajectories, and the expert provides the correct action for each visited state. These new state-action pairs are aggregated into a growing dataset, and the next policy is trained on it. Because the dataset now includes states encountered by the learned policy, the mismatch between the training and execution distributions is gradually reduced. The full procedure is given in Program 1 [10].

```
Initialize dataset 𝒟 ← ∅.
Initialize π̂_1 to any policy in ∏.
For i=1 to N do
   Let π_i = β_i π* + (1 − β_i) π̂_i.
   Sample T-step trajectories using π_i.
```

```
   Get dataset D_i = {(s, π*(s))} of visited states by π_i and actions given by
   expert.
   Aggregate datasets: 𝒟 ← 𝒟 ∪ D_i.
   Train classifier π̂_{i+1} on 𝒟.
End for
Return best π̂_i on validation.
```

Program 1. *Dagger Algorithm*

Here β_i is a mixing coefficient that determines the proportion of expert and learned policy actions used during data collection. It is typically decayed from 1 to 0 across iterations, so that early rounds rely on the expert while later rounds collect data primarily from the learned policy [10].

3 PROBLEM FORMULATION

This chapter formulates the cooperative transport problem addressed in this thesis. We consider groups of unicycle robots that collaboratively carry rigid objects to goal positions in a shared environment. Each object is carried by one group of robots. The two robots in a group must keep a fixed distance from each other and stay aligned face-to-face, so that they hold the object stably during transport. When several groups move at the same time in the shared map, they must also avoid colliding with robots from other groups.

Section 3.1 introduces the system model, including the unicycle kinematics, the group structure, and the time discretization scheme. Section 3.2 specifies the constraints: velocity bounds, intra-group formation, inter-group collision avoidance, and goal-reaching accuracy. Section 3.3 states the assumptions adopted in this work. Section 3.4 gives the formal problem statements for the single-group and multi-group cases.

3.1 System Model

Consider a 2D environment with N groups of robots. Each group $m \in \{1, 2, \dots, N\}$ consists of two unicycle robots, indexed by $i \in \{1, 2\}$. The two robots in group cooperatively transport a rigid object of fixed length L_o, which they grasp rigidly at both ends.

The state of robot i in group m at time t is defined as:

$$\mathbf{x}_{m,i}(t) = [p_{m,i,x}(t),\ p_{m,i,y}(t),\ \theta_{m,i}(t)]^T \tag{3.1}$$

Where the $p_{m,i,x}(t)$, $p_{m,i,y}(t) \in \mathbb{R}$ denote the planar position of the robot and $\theta_m(t) \in [-\pi, \pi]$ is its heading angle. The control input of each robot is

$$\mathbf{u}_{m,i}(t) = [v_{m,i}(t),\ \omega_{m,i}(t)]^T, \tag{3.2}$$

where $v_{m,i}(t)$ is the linear velocity and $\omega_{m,i}(t)$ is the angular velocity. Each robot follows the unicycle kinematics [1]

$$\dot{\mathbf{x}}_{m,i}(t) = \begin{bmatrix} \dot{p}_{m,i,x}(t) \\ \dot{p}_{m,i,y}(t) \\ \dot{\theta}_{m,i}(t) \end{bmatrix} = \begin{bmatrix} \cos\theta_{m,i}(t) & 0 \\ \sin\theta_{m,i}(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{m,i}(t) \\ \omega_{m,i}(t) \end{bmatrix} \tag{3.3}$$

For the simulation, we integrate this kinematics using the midpoint Euler scheme with time step Δt:

$$\begin{aligned} \theta_{m,i}^{mid} &= \theta_{m,i}(t) + \tfrac{1}{2}\omega_{m,i}(t)\Delta t, \\ \mathbf{x}_{m,i}(t + \Delta t) &= \mathbf{x}_{m,i}(t) + \begin{bmatrix} v_{m,i}(t)\cos\theta_{m,i}^{mid} \\ v_{m,i}(t)\sin\theta_{m,i}^{mid} \\ \omega_{m,i}(t) \end{bmatrix} \Delta t \end{aligned} \tag{3.4}$$

Each group m is assigned a static goal position $g_m = [x_{m,g}, y_{m,g}]^T \in \mathbb{R}^2$. The goal is the desired final position of one end of the transported object. We associate this end with robot $i = 1$ in group m. The goal reaching condition is then:

$$\lim_{t \to t_f} \left\| p_{m,1}(t) - g_m \right\| \leq \epsilon_{goal} \tag{3.5}$$

Where $\epsilon_{goal} > 0$ is the defined goal-reaching tolerance.

3.2 System Constraints

This section specifies the constraints that the robots need to satisfy throughout the transport process.

Each robot's control input is bounded by physical actuator limits. For all $m \in \{1, 2, \dots, N\}$, $i \in \{1,2\}$, and $t \in [0, t_f]$:

$$\left| v_{m,i}(t) \right| \leq V_{max},\ \left| \omega_{m,i}(t) \right| \leq \Omega_{max} \tag{3.6}$$

Within each group m, the two robots cooperatively grasp a rigid object of length L_o at its two ends. To keep the object held during the motion, two coupled constraints must hold. First, the distance between the two robots must stay close to L_o. A small deviation ϵ_{form} is allowed to account for the elastic compliance of the robotic arms:

$$\left| \left\| p_{m,1}(t) - p_{m,2}(t) \right\| - L_o \right| < \epsilon_{form} \tag{3.7}$$

Second, the two robots must stay in a face-to-face heading configuration. Their headings must differ by π, up to a tolerance ϵ_{face} that accounts for the wrist-twisting range of the grasping:

$$\left| \mathrm{wrap}\left(\theta_{m,1}(t) - \theta_{m,2}(t) \right) - \pi \right| < \epsilon_{face} \tag{3.8}$$

where $\mathrm{wrap}(\cdot)$ is the standard angle-wrapping function that keeps the heading angle difference of two robots within $(-\pi, \pi]$.

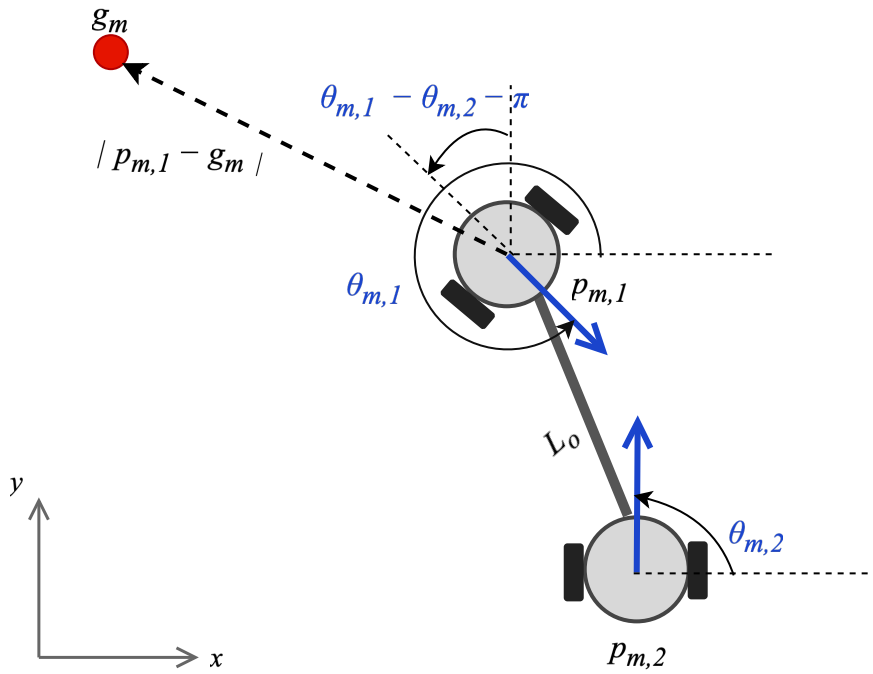


Figure 2. *Geometry of a single group during transport.*

Figure 2 shows the geometry of a single group. The two robots, located at $p_{m,1}$ and $p_{m,2}$ grasp the object of length L_o at its two ends. The headings are $\theta_{m,1}$ and $\theta_{m,2}$ respectively. The goal g_m is associated with robot $i = 1$, and $\|p_{m,1} - g_m\|$ denotes its distance to the goal.

When multiple groups operate in a shared map, robots from different groups can not collide. For all groups $m \neq n$, all $i, j \in \{1,2\}$, and all $t \in [0, t_f]$:

$$\|p_{m,i}(t) - p_{n,j}(t)\| \geq 2r_{safe} \tag{3.9}$$

Where r_{safe} is the safety radius around each robot. Besides, each group m must deliver the object to its assigned goal position g_m. As noted above, this end is associated with robot $i = 1$, making the terminal condition is

$$\lim_{t \to t_f} \|p_{m,1}(t) - g_m\| \leq \epsilon_{goal} \tag{3.10}$$

3.3 Assumptions

The following assumptions are adopted throughout this thesis. They make the cooperative transport problem approachable while preserving its essential structure.

We first assume that the transported object is rigid and has a fixed length L_o. The two robots in each group keep a grasp on the object at its two ends throughout the motion, with no slip or release. The pose of the object is therefore fully determined by the positions of the grasping robots. All motion takes place on a flat 2D environment.

Next, each robot has full access to its own state $\mathbf{x}_{m,i}(t)$, without noise or delay. Within each group, the two robots share their states information instantaneously and reliably. Across groups, only robot ($i = 1$) of each group communicates. Its state $\mathbf{x}_{m,1}(t)$ of is shared with robot $i = 1$ of another group $m \neq n$ only when the two are within a communication range r_{comm} of each other. Each robot can sense the relative position of its nearest robot from another group within a sensing range regardless of role, which we also set to r_{comm}. All communication and sensing within this range are treated as instantaneous and lossless.

Finally, the goal positions g_m are given in advance and stay fixed throughout the transport process. The control inputs $\mathbf{u}_{m,i}(t)$ are subject to the magnitude bounds introduced in Section 3.2.

3.4 Problem Statement

Building upon previous sections, the cooperative transport problems are stated as follows.

Problem 1 (Single-group cooperative transport):

Consider a single group $(N = 1)$ of unicycle robots indexed by $i \in \{1, 2\}$, with given initial states $\{\mathbf{x}_{1,i}(0)\}_{i\in\{1,2\}}$ and a static goal position g_1. Find a control policy π: $\{\mathbf{x}_{1,i}(t)\}_{i\in\{1,2\}} \rightarrow \{\mathbf{u}_{1,i}(t)\}_{i\in\{1,2\}}$, such that, for the trajectory produced by the unicycle dynamics (3.3) under the policy π:

- the input bounds (3.6) hold for all $t \in [0, t_f]$ and $i \in \{1, 2\}$;
- the intra-group formation constraints (3.7) - (3.8) hold for all $t \in [0, t_f]$;
- the goal-reaching condition (3.10) holds at $t = t_f$.

Problem 2 (Multi-group cooperative transport):

Consider $N > 1$ groups of unicycle robots, indexed by $m \in \{1, \ldots, N\}$ and $i \in \{1, 2\}$, with given initial states $\{\mathbf{x}_{m,i}(0)\}_{m,i}$ and respective static goal positions $\{g_m\}_{m=1}^{N}$. Find a control policy $\pi : \{x_{m,i}(t)\}_{m,i} \rightarrow \{u_{m,i}(t)\}_{m,i}$, such that, for the trajectory generated by the unicycle dynamics (3.3) under the policy π:

- the input bounds (3.6) hold for all $t \in [0, t_f]$, all m, and all i;

- the intra-group formation constraints (3.7)-(3.8) hold for all $t \in [0, t_f]$ and all m;
- the inter-group collision-avoidance constraint (3.9) holds for all $t \in [0, t_f]$ and all distinct groups $m \neq n$;
- the goal-reaching condition (3.10) holds at $t = t_f$ for every m.

Problem 1 sets up the baseline single-group case. Problem 2 extends it to the multi-group case, which adds the challenge of inter-group coordination. The rest of this thesis develops a GNN-based approach to Problem 2, with Problem 1 serving as a building block in both the algorithm design and the simulation evaluation.

4 PROPOSED METHOD

The proposed method treats the multi-group cooperative transport problem as a graph-structured control task. As introduced in Chapter 2, a GNN operates on graph-structured data by passing messages between nodes along edges. This structure fits the system described in Chapter 3 naturally: each robot is a node, the intra-group grasping coupling and inter-group communication are encoded as edges, and the control command for each robot is produced from its node embedding after several rounds of message passing.

The overall framework is illustrated in Figure 3. It consists of two loops. The inference loop runs in closed loop. At each time step, the current robot states are used to build a graph $G(t)$, which is passed through the GNN to produce control commands $\mathbf{u}_{m,i} = (v_{m,i}, \omega_{m,i})$ for all robots. These commands are applied to the simulation, and the updated states are used to rebuild the graph for the next step.

The training loop operates offline. The GNN is trained on a dataset D that combines expert demonstrations generated by the QP-CBF controller and correction samples collected through DAgger. At each training iteration, graphs drawn from D are passed through the GNN to compute a composite loss. This loss combines behavioral cloning (BC) of the expert action with physics-based self-supervision terms for goal-reaching, formation maintenance, and inter-group safety. The loss is backpropagated through the network, and its parameters are updated. After training converges, the dataset is expanded with further data through DAgger, and training repeats.

Section 4.1 describes how the multi-group system is represented as a graph, including the node set, the edge set, and the node features. Section 4.2 presents the network architecture, covering the node encoder, the GAT backbone, and the action heads. Section 4.3 defines the loss function, which combines behavioral cloning with physics-based self-supervision terms. Section 4.4 describes the training procedure, including data collection and the DAgger process.

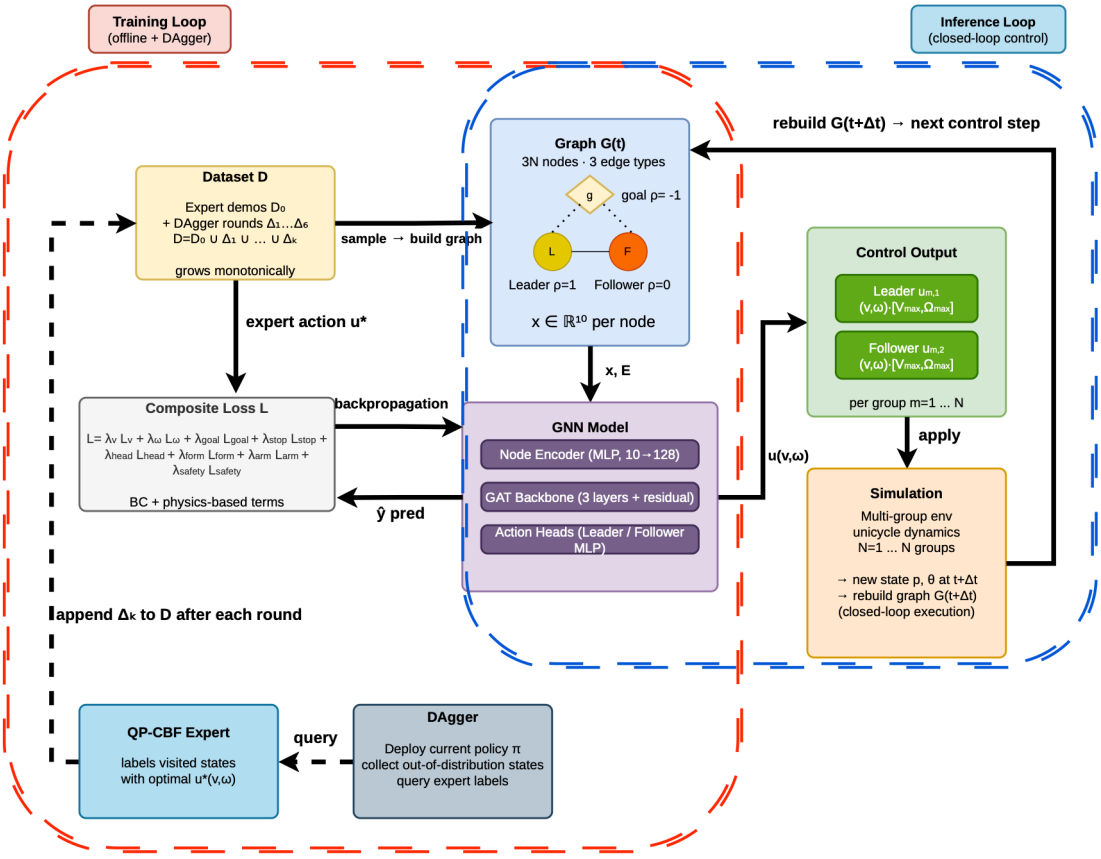


Figure 3. *Overview of the proposed GNN-based control framework, showing the inference loop and the training loop.*

4.1 Graph Construction

We model the multi-group cooperative transport system at each time step t as a graph $G(t) = (V, E(t), X(t))$, where V denotes the node set, $E(t)$ the edge set, and $X(t)$ the node feature matrix. The construction is presented for the general case of $N \geq 1$ groups. The single-group scenario is then the special case $N = 1$, in which the inter-group edge set defined below is empty.

4.1.1 Node Set

Each robot is associated with a node, and each group's goal position is associated with an additional node. The full node set is $V = \bigcup_{m=1}^{N} \{(m, 1), (m, 2), g_m\}$.

The face-to-face heading constraint introduced in Section 3.2 implies that, throughout the transport, one robot in each group travels backward (relative to its body frame) while the other travels forward. We use this geometric asymmetry as the basis for an asymmetric task decoupling within the group. The backward-travelling robot, indexed by $i = 1$,

navigates toward the goal and handles inter-group collision avoidance. The forward-travelling robot, indexed by $i = 2$, primarily maintains the formation behind its groupmate. Both robots sense nearby robots from other groups for collision avoidance. We refer to robot $i = 1$ as the leader node and robot $i = 2$ as the follower node of group m. The leader-follower distinction is represented by a role-indicator entry in the node feature vector, defined in 4.1.3.

The goal g_m is included as a separate node. This allows goal information to propagate through the message-passing mechanism, along the edges that connect the robots. In total, the graph has $|V| = 3N$ nodes, and the types are leader, follower, and goal.

4.1.2 Edge Set

The edge set is divided into three types: intra-group edges, goal edges, and inter-group edges. The first two are fixed throughout the motion, while the third varies with time according to the relative positions of the communicating robots.

Intra-group edges encode the rigid grasping coupling between the two robots within each group:

$$E_{intra} = \{(m, 1), (m, 2) : m \in \{1, \dots, N\}\}, \tag{4.1}$$

Goal edges connect both robots in each group to the corresponding goal node, so that goal information is accessible within each group's subgraph:

$$E_{goal} = \{(m, i), g_m : m \in \{1, \dots, N\}, i \in \{1,2\}\}, \tag{4.2}$$

Inter-group edges are constructed based on the distance between robots, a common strategy for defining neighborhoods in multi-robot graph representations[5]. These edges encode the communication described in Section 3.2, which restricts inter-group information exchange to the robot indexed by $i = 1$ in each group. Among these robots, we connect only those that lie within a fixed communication radius r_{comm} of one another. Groups farther apart than r_{comm} have no immediate collision risk and therefore do not need to exchange information:

$$E_{inter}(t) = \{(m, 1), (n, 1) : m \neq n, \left\|p_{m,1}(t) - p_{n,1}(t)\right\| < r_{comm}\}, \tag{4.3}$$

These edges represent communication links, through which leaders exchange their state representations during message passing. The connectivity is distance-based and therefore time-varying, since the inter-robot distances change along the trajectory. Restricting communication to nearby leaders keeps the message-passing neighborhood compact and its computational cost manageable as N grows. Figure 4 illustrates this construction.

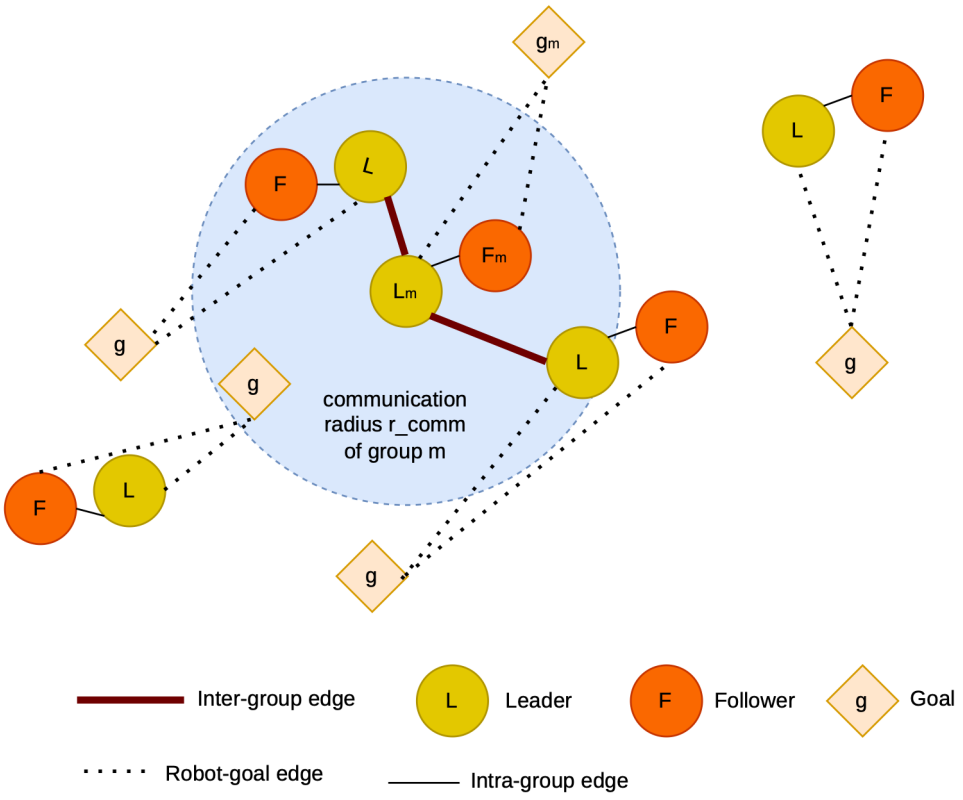


Figure 4. *Inter-group communication based on the communication radius.*

The complete edge set is $E(t) = E_{intra} \cup E_{goal} \cup E_{inter}(t)$. All edges are bidirectional. The robot indexed by $i = 2$ in each group, together with the goal nodes, participate only in intra-group interactions and have no inter-group connections. When $N = 1$, the inter-group edge set $E_{inter}(t)$ is empty, and the graph reduces to a fully connected triangle on the two robots and the goal node.

In this work, edges carry only adjacency information and no explicit edge feature vectors are constructed. The attention weights used in message passing are computed directly from the node features.

4.1.3 Node Features

To allow uniform processing by the GNN, all three node types (leader, follower, and goal) share a feature vector of dimension $d = 10$. Although the semantic content of the features differs across node types, we pad and align all node features to a 10-dimensional space. During the message passing, this lets a single shared parameterization to be applied to all nodes.

The features are grouped into five categories: goal-relative information, heading information, intra-group formation information, inter-group observation, and a role indicator.

Each entry is normalized by a scale constant, which keeps all features in a comparable numerical range and stabilize training.

The feature vector $\mathbf{x}_{m,i}(t)$ for robot (m,i) at time t is

$$\mathbf{x}_{m,i}(t) = [\frac{\Delta x_{m,i}^{g}}{D_{max}}, \frac{\Delta y_{m,i}^{g}}{D_{max}}, \frac{\theta_{m,i}}{\pi}, \frac{d_m}{D_{max}}, \frac{\alpha_m}{\pi}, \frac{L_o}{D_{max}}, \frac{d_m - L_o}{D_{form}}, \frac{obs_{m,i}^{x}}{r_{comm}}, \frac{obs_{m,i}^{y}}{r_{comm}}, \rho_i]^T, \tag{4.4}$$

where:

1. $\Delta x_{m,i}^{g}$, $\Delta y_{m,i}^{g}$ are the planar displacements from the leader robot (m,i) to the goal g_m;
2. $\theta_{m,i}$ is the heading of robot (m,i);
3. $d_m = \|p_{m,1}(t) - p_{m,2}(t)\|$ is the current inter-robot distance in group m;
4. $\alpha_m = \mathrm{wrap}(\theta_{m,1} - \theta_{m,2})$ is the relative angle between the two robots' headings;
5. $obs_{m,i}^{x}$, $obs_{m,i}^{y}$ are the planar displacements from robot (m,i) to its nearest robot in a different group, clipped to a magnitude of r_{comm};
6. $\rho_i \in \{0,1\}$ is the role indicator, with $\rho_1 = 1$ for the leader and $\rho_2 = 0$ for the follower.
7. Normalization constant D_{max} is the workspace diameter.
8. D_{form} is the formation-error scale.
9. L_o is the object length.
10. r_{comm} is the communication radius.

The feature vector $\mathbf{x}_{g_m}(t)$ of goal node g_m is

$$\mathbf{x}_{g_m}(t) = [\widehat{\Delta x_{m,1}^{g}}, \widehat{\Delta y_{m,1}^{g}}, \sin(\theta_{m,1}), \cos(\theta_{m,1}), 0,0,0,0,0, \rho_g]^T, \tag{4.5}$$

where $(\widehat{\Delta x_{m,1}^{g}}, \widehat{\Delta y_{m,1}^{g}})$ is the unit vector pointing from the leader of group m toward the goal g_m, and $\sin(\theta_{m,1})$, $\cos(\theta_{m,1})$ encode the leader's heading. The role indicator $\rho_g = -1$ distinguishes the goal from leader $(\rho_1 = 1)$ and follower $(\rho_2 = 0)$ nodes. The remaining five entries are padded as zero, since the goal node is static and carries no formation, inter-group observation, or kinematic information of its own.

Table 4.1 summarizes the node feature entries for three node types. Their specific values, together with the choice of feature clipping, are detailed in Chapter 5.

Table 4.1. *Node feature entries*

index	Leader$(i = 1)$	Follower$(i = 2)$	Goal(g_m)
1	$\Delta x_{m,1}^{g}/D_{max}$	$\Delta x_{m,2}^{g}/D_{max}$	$\widehat{\Delta x_{m,1}^{g}}$
2	$\Delta y_{m,1}^{g}/D_{max}$	$\Delta y_{m,2}^{g}/D_{max}$	$\widehat{\Delta y_{m,1}^{g}}$
3	$\theta_{m,1}/\pi$	$\theta_{m,2}/\pi$	$\sin(\theta_m, 1)$
4	d_m/D_{max}	d_m/D_{max}	$\cos(\theta_m, 1)$
5	α_m/π	α_m/π	0
6	L_o/D_{max}	L_o/D_{max}	0
7	$(d_m - L_o)/D_{form}$	$(d_m - L_o)/D_{form}$	0
8	$obs_{m,1}^{x}/r_{comm}$	$obs_{m,2}^{x}/r_{comm}$	0

9	$obs^{y}_{m,1}/r_{comm}$	$obs^{y}_{m,2}/r_{comm}$	0
10	$+1$	0	-1

Each feature category is chosen to give the network the information needed for specific aspects of the task. The goal-relative information uses the displacement from each robot to its goal instead of the robot's absolute position. This makes the policy invariant to where a group starts or goal places in the workspace, so a policy trained on one set of start and goal positions transfers to others. The heading $\theta_{m,i}$ tells each robot which way it is facing, which it needs to turn toward and drive to the goal. The intra-group formation information: inter-robot distance d_m and the relative heading α_m, makes the two robots in a group keep the fixed distance and the face-to-face configuration required to hold the object. The displacement to the nearest robot of another group for inter-group observation, gives each robot a local sense of nearby groups for collision avoidance. The role indicator ρ_i allows the network to tell leaders, followers, and goal nodes apart, so that a single shared model can assign the correct behaviour to different node type.

4.2 Network Architecture

As shown in Figure 5, the proposed network maps the graph $G(t)$ to robot control commands in three stages. First, a shared node encoder projects the 10-dimensional node features into a common a 128-embedding embedding $h^{(0)}$. Second, a three-layer GAT backbone refines the embeddings through multi-hop message passing, with a residual connection that adds $h^{(0)}$ to the final layer output to produce h^{out}. Third, two role-specialized action heads map each robot's embedding to the scaled kinematic commands.

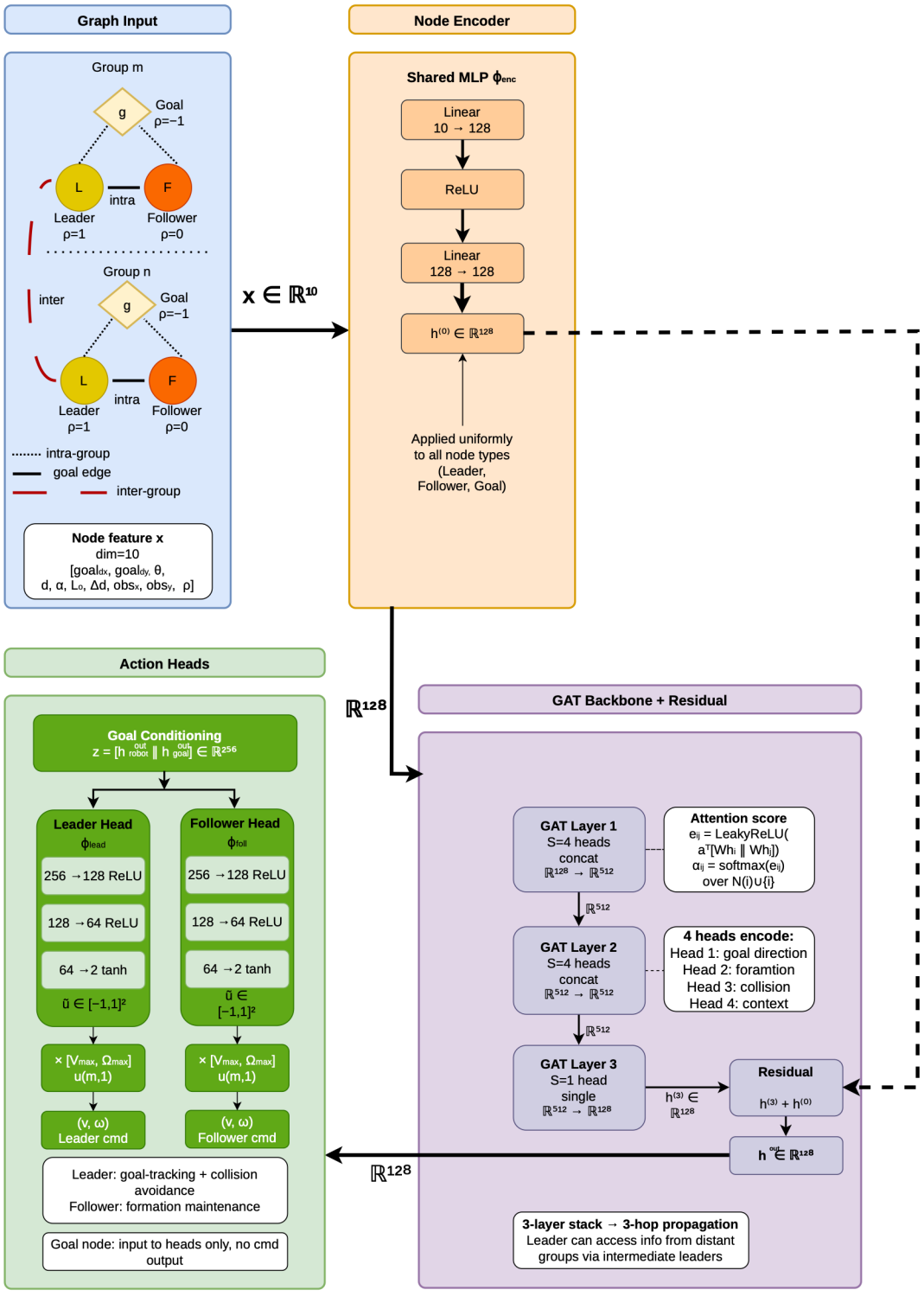


Figure 5. *GNN Architecture.*

4.2.1 Node Encoder

The 10-dimensional raw input feature $\mathbf{x} \in \mathbb{R}^{10}$ of every node is first projected into a hidden embedding of dimension $d_h = 128$ by a shared two-layer multilayer perceptron (MLP):

$$h^{(0)} = \phi_{enc}(\mathrm{x}) = W_2\sigma(W_1\mathrm{x} + b_1) + b_2 \quad (4.6)$$

where $\sigma(\cdot) = \max(0,\cdot)$ denotes the Rectified Linear Unit (ReLU) activation, and $W_1 \in \mathbb{R}^{128\times10}$, $W_2 \in \mathbb{R}^{128\times128}$ are the layer weight matrices. The same encoder ϕ_{enc} is applied to leader, follower, and goal nodes to keep with the uniform feature-vector design introduced in Section 4.1.3. The encoder produces an initial embedding $h_i^{(0)} \in \mathbb{R}^{128}$ for every node $i \in V$.

4.2.2 GATs

The initial node embeddings $h_i^{(0)}$ are refined through three stacked GATs [9] with multi-head attention, followed by a residual connection from the encoder output. The neighborhood of node i in the graph is $N(i) = \{j : (i,j) \in E(t)\}$, which evolves over time because of the distance-gated inter-group edges defined in Section 4.1.2.

We adopt attention-based aggregation rather than the uniform aggregation of standard GCNs (Graph Convolutional Networks) [11] because the three categories of edges in our graph carry different information. The weight a node should be placed on different edge depends on its role and on the surrounding context. The learnable attention lets this weight be inferred from data. The choice of three layers allows information to propagate up to three hops in a single forward pass. Combined with the distance-gated inter-group, this lets a leader indirectly access information from distant groups through intermediate leader nodes.

At layer $l \in \{1,2,3\}$, each node i aggregates information from its 1-hop neighbors $N(i)$ and from itself, through S parallel attention heads:

$$\begin{aligned} h_i^{(l)} &= \Big\|_{s=1}^{S}\sigma\Big(\sum\nolimits_{j\in N(i)\cup\{i\}} \alpha_{ij}^{(l,s)} W^{(l,s)} h_j^{(l-1)}\Big) \\ \alpha_{ij}^{(l,s)} &= \frac{\exp(e_{ij}^{(l,s)})}{\sum_{m\in N(i)\cup\{i\}}\exp(e_{im}^{(l,s)})} \\ e_{ij}^{(l,s)} &= \mathrm{LeakyReLU}(a^{(l,s)T}[W^{(l,s)}h_i^{(l-1)} \,\|\, W^{(l,s)}h_j^{(l-1)}]) \end{aligned} \quad (4.7)$$

Here $h_j^{(l-1)}$ is node j's embedding from the previous layer, and $h_i^{(l)}$ is node i's updated embedding at current layer. The symbol ‖ denotes vector concatenation, and $\sigma(\cdot)$ is the ReLU activation.

The attention weight $\alpha_{ij}^{(l,s)} \in [0,1]$ specifies how much node i listens to neighbor j under head s in layer l. It is obtained by applying a softmax over the unnormalized attention scores within the self-augmented neighborhood $N(i) \cup \{i\}$, so that for each fixed (i,s) the weights sum to one. The unnormalized score $e_{ij}^{(l,s)}$ measures how relevant neighbor j is to node i under head s. To computed it, both embeddings $h_i^{(l-1)}$ and $h_j^{(l-1)}$ are first

transformed by the head's weight matrix $W^{(l,s)}$. The two transformed vectors are concatenated, a learnable vector $a^{(l,s)}$ maps the result to a single number, and a LeakyReLU is applied. The same matrix $W^{(l,s)}$ is used both here and in the aggregation above, where the neighbor features are combined.

The weights are not fixed. Because the attention weight $\alpha_{ij}^{(l,s)}$ is computed from the embeddings of node i and node j, the network can learn to give more weight to the neighbors that carry the most useful information for each role and context.

Following the multi-head formulation introduced in Section 2.2, we use of $S = 4$ heads in the first two layers and $S = 1$ in the final layer. Multiple heads let the network attend to different aspects of each robot's context in parallel, such as its goal direction, its intra-group formation, and nearby groups.

The three layers are configured as follows:

Layer 1: $S = 4$, concat $\rightarrow$ $\mathbb{R}^{128} \rightarrow \mathbb{R}^{512}$;
Layer 2: $S = 4$, concat $\rightarrow$ $\mathbb{R}^{512} \rightarrow \mathbb{R}^{512}$;
Layer 3: $S = 1$, single $\rightarrow$ $\mathbb{R}^{512} \rightarrow \mathbb{R}^{128}$.

The first two layers concatenate the four heads into a 512 dimensions representation, preserving each head's specialized signal across layers. The third layer uses a single head, compressing the representation back to 128 dimensions to produce a compact embedding for the downstream action heads.

Repeated message passing through deep GNN stacks tends to homogenize node embeddings. To mitigate it, we add a residual connection from the encoder output to the final GAT layer [12]: $h_i^{out} = h_i^{(3)} + h_i^{(0)} \in \mathbb{R}^{128}$. This connection helps each node get direct access to its own encoded features in the final embedding h_i^{out}.

4.2.3 Action Heads

The refined node embeddings produced by the GAT backbone are mapped into robot control commands by two role-specialized MLPs: a leader head ϕ_{lead} and a follower head ϕ_{foll}. We adopt separate heads instead of a single shared head, because the two roles carry different control tasks. The leader tracks the goal and avoids inter-group collisions, while the follower maintains the formation behind the leader.

Goal-conditioned input

Each action head takes the concatenation of two final embeddings as input: the robot's own embedding $h_{m,i}^{out}$ and the embedding $h_{g_m}^{out}$ of its corresponding goal node:

$$z_{m,i} = [h_{m,i}^{out} \parallel h_{gm}^{out}] \in \mathbb{R}^{2d_h} \quad (4.8)$$

Where d_h=128 and || denotes concatenation, the concatenation making $z_{m,i} \in \mathbb{R}^{256}$. Although goal information already propagates through the GAT layers through the goal-node edges, this explicit concatenation gives each action head a direct view of the goal.

Per-robot control prediction

The bounded output $\tilde{u}_{m,i}$ of each head is produced by a three-layer MLP followed by a tanh nonlinearity,

$$\tilde{u}_{m,i} = \tanh(\phi_{r_i}(z_{m,i})) \in [-1,1]^2 \quad (4.9)$$

where $r_i \in \{lead, foll\}$ selects the head based on the robot's role. The MLP ϕ_{r_i} has hidden widths 128 and 64 with ReLU activations:

$$\phi_{r_i}: \mathbb{R}^{256} \xrightarrow{\text{Linear+ReLU}} \mathbb{R}^{128} \xrightarrow{\text{Linear+ReLU}} \mathbb{R}^{64} \xrightarrow{\text{Linear}} \mathbb{R}^2$$

The two heads share this structure MLP but keep independent parameters.

The tanh-bounded output $\tilde{u}_{m,i} \in [-1,1]^2$ is mapped to the physical control input by:

$$u_{m,i}(t) = \begin{bmatrix} V_{max} \\ \Omega_{max} \end{bmatrix} \odot \tilde{u}_{m,i} \quad (4.10)$$

where $\odot$ denotes element-wise multiplication. Goal nodes do not produce any control commands; their final embeddings serve only as inputs to the action heads of the corresponding group's robots.

4.2.4 Code Summary

Program 2 summarizes the complete forward pass of the network in pseudocode. This procedure is run at every control step during deployment and training.

```
Input:  graph G(t) = (V,E(t),X(t))
Output: control commands {u_{m,i}(t)}

// Node encoding
for each node v∈V do
   h_v^(0) ← φ_enc(x_v)
end for

// GAT backbone with residual
H^(1) ← GAT (H^(0), E; S = 4, concat) // → ℝ^{|V|×512}
H^(2) ← GAT (H^(1), E; S = 4, concat) // → ℝ^{|V|×512}
H^(3) ← GAT (H^(2), E; S = 1, single) // → ℝ^{|V|×128}
H^out ← H^(3) + H^(0)                 // residual

// Action heads
for each group m ∈ {1, …, N} do
   for each robot i ∈ {1,2} do
```

```
            z_{m,i} ← [h_{m,i}^out ‖ h_{g_m}^out]
            if i = 1 then
                  ũ_{m,i} ← tanh(φ_lead(z_{m,i}))
            else
                  ũ_{m,i} ← tanh(φ_foll (z_{m,i}))
            end if
            u_{m,i}(t) ← [V_max, Ω_max]^T ⊙ ũ_{m,i}
      end for
   end for

   return {u_{m,i}(t): m∈{1…,N},i∈{1,2}}
```

Program 2. *Forward pass of the proposed GNN controller.*

The pseudocode lists the three stages in the order they run: every node is first encoded, the embeddings are then refined by the three GAT layers with the residual connection, and finally each robot's embedding is combined with its goal embedding and passed through its role-specific head to produce the command. Goal nodes are encoded but produce no command of their own.

4.3 Loss Function

The standard practice in supervised learning is to minimize the mean squared error (MSE) between the predicted action and the expert action. While this pure behavioral cloning objective is simple, it suffers from compounding errors at deployment. A small deviation from the expert trajectory pushes the system into out-of-distribution states, where the network's predictions degrade fast. Because our expert dataset cannot cover every configuration the multi-group system may visit, this problem will damage in our setting. We address it in two complementary ways: the DAgger procedure, which collects expert labels on the states the learner visits, and the physics-based loss terms described below.

We adopt a composite loss that augments the standard BC with physics-based self-supervision losses. A differentiable unicycle kinematic predictor is embedded directly into the loss, so that each predicted action can be rolled forward by one time step Δt to obtain the predicted next state $(x_{next}, y_{next}, \theta_{next})$. The self-supervision terms are then defined on these predicted next states. This turns abstract action errors into concrete physical consequences, such as distance to goal after one step, the formation deviation, or the collision risk.

4.3.1 Kinematic Predictor

All physics-based losses are defined on the predicted next state of each robot, not on its predicted action directly. The network outputs an action $(v_{m,i}, \omega_{m,i})$ at time t. We roll the

state forward over a prediction horizon T_{pred} using the same midpoint Euler discretization as in Section 3.1, applied repeatedly over T_{pred}.

$$\hat{\theta}_{m,i}^{mid} = \theta_{m,i}(t) + \frac{1}{2}\omega_{m,i}(t)T_{pred},$$

$$\begin{bmatrix} \hat{p}_{m,i,x}(t+T_{pred}) \\ \hat{p}_{m,i,y}(t+T_{pred}) \\ \hat{\theta}_{m,i}(t+T_{pred}) \end{bmatrix} = \begin{bmatrix} p_{m,i,x}(t) \\ p_{m,i,y}(t) \\ \theta_{m,i}(t) \end{bmatrix} + \begin{bmatrix} v_{m,i}(t)\cos\hat{\theta}_{m,i}^{mid} \\ v_{m,i}(t)\sin\hat{\theta}_{m,i}^{mid} \\ \omega_{m,i}(t) \end{bmatrix} T_{pred} \tag{4.11}$$

The predicted next state at $t+T_{pred}$, denoted ($\hat{p}_{m,i,x}(t+T_{pred}), \hat{p}_{m,i,y}(t+T_{pred}), \hat{\theta}_{m,i}(t+T_{pred})$), carries a hat to distinguish it from the realized next state in simulation. Because the predicted next state is a differentiable function of the predicted action, the self-supervision losses defined on it produce gradients on the action prediction during backpropagation.

Here T_{pred} acts as a prediction horizon. It should not be too small. Over a very short horizon, the robot barely moves, so the physics-based losses change little, and their gradients become too weak to train on. A longer horizon makes the effect of each action clearer and gives a stronger training signal.

4.3.2 Behavioral Cloning Loss

For each training sample with an available expert action $\mathbf{u}_{m,i}^* = (v_{m,i}^*, \omega_{m,i}^*)$, we penalize the MSE between the predicted action and the expert action, computed separately for the linear and angular velocities:

$$L_v = \frac{1}{|B|}\sum_{(m,i)\in B}(v_{m,i} - v_{m,i}^*)^2, \qquad L_\omega = \frac{1}{|B|}\sum_{(m,i)\in B}(\omega_{m,i} - \omega_{m,i}^*)^2 \tag{4.12}$$

where B is the set of all robot samples in the training batch and $|B|$ its size. The two components are kept separate so that the linear and angular velocity errors can be weighted independently.

4.3.3 Navigation Losses

The navigation losses shape the leader's motion toward the goal. They have three components: a goal-distance loss that drives the leader to the goal, a heading loss that orients the leader, and a stop loss that combines velocity tracking with a parking lock. All three are computed from the predicted next state.

We first define a parking indicator that separates leaders still in transit and those that have arrived:

$$\hat{d}_m = \left\|\hat{p}_{m,1}(t+T_{pred}) - g_m\right\|, \qquad 1_{park}(m) = 1[\hat{d}_m < \varepsilon_{park}] \tag{4.13}$$

Where $\hat{d}_m$ is the predicted leader-to-goal distance and ε_{park} is the parking tolerance. $1[\cdot]$ is the indicator function, equal to 1 when its condition holds and 0 otherwise.

Goal Distance Loss

For leaders that have not yet parked, this loss penalizes the predicted distance to the goal:

$$L_{goal} = \frac{1}{|P|}\sum_{m\in P}(1 - 1_{park}(m))\hat{d}_m \tag{4.14}$$

where P is the set of group indices in the training batch. The parking indicator switches off this term once the leader is within ε_{park} of its goal, which prevents the loss from pushing the leader past the goal.

Heading Loss

The face-to-face heading constraint requires the leader to travel backward in its body frame, so its heading should point opposite to the direction of travel. Let

$$\beta_m = \text{atan2}(g_{m,y} - \hat{p}_{m,1,y}, g_{m,x} - \hat{p}_{m,1,x}) \tag{4.15}$$

be the angle of the leader-to-goal vector at the predicted next position. The desired leader heading is then $\beta_m + \pi$, and the heading loss penalizes the wrapped deviation from this target:

$$L_{head} = \frac{1}{|P|}\sum_{m\in P}(1 - 1_{park}(m))\left|wrap(\hat{\theta}_{m,1} - \beta_m - \pi)\right| \tag{4.16}$$

The target angle β_m is detached from the computational graph, so that gradients flow only into the predicted heading $\hat{\theta}_{m,1}$. As with the goal-distance loss, the parking indicator deactivates this term once the leader has arrived.

Stop Loss

The stop loss combines two roles. While the leader is in transit, both robots track a desired speed profile. Once the leader is parked, it stops. The follower stops only when the leader is parked and the formation is also correct.

For the velocity-tracking component, we adopt the speed profile from the QP-CBF expert[1]:

$$v_{cmd}(\hat{d}_m) = V_{max}(1 - e^{-k_v\hat{d}_m}) \tag{4.17}$$

with $k_v > 0$, controlling the rate of deceleration as the leader approaches the goal. A larger k_v makes the leader slow down earlier; a smaller k_v keeps it at higher speed until closer to the goal. The leader travels backward and the follower forward, making opposite-signed target velocities:

$$v_{m,1}^{tgt} = -v_{cmd}(\hat{d}_m), \qquad v_{m,2}^{tgt} = +v_{cmd}(\hat{d}_m) \tag{4.18}$$

The velocity-tracking loss penalizes deviation from these targets while the leader is not parked:

$$L_{vel} = \frac{c_{vel}}{|P|} \sum_{m \in P} (1 - 1_{park}(m)) \sum_{i \in \{1,2\}} (v_{m,i} - v_{m,i}^{tgt})^2 \tag{4.19}$$

where c_{vel} is a scale factor.

The parking-lock component locks the robots once the goal is reached. The leader is locked whenever its parking indicator is active. The follower is locked only when the leader is parked, and the formation distance is within a tolerance ε_{form} of the object length L_o:

$$1_{park}^{F}(m) = 1_{park}(m) \cdot 1[\left| \left\| \hat{p}_{m,1} - \hat{p}_{m,2} \right\| - L_o \right| < \varepsilon_{form}] \tag{4.20}$$

The lock loss penalizes any non-zero predicted velocity while the corresponding indicator is active:

$$L_{lock} = \frac{c_{lock}}{|P|} \sum_{m \in P} [1_{park}(m) v_{m,1}^2 + 1_{park}^{F}(m) v_{m,2}^2] \tag{4.21}$$

where c_{lock} is a scale factor. The follower's lock is enabled only when the formation is correct. This prevents the follower from freezing far away from the leader. If the formation is violated when the leader parks, the follower keeps moving until the formation is restored. The total stop loss is $L_{stop} = L_{vel} + L_{lock}$.

4.3.4 Formation Loss

Two losses enforce the intra-group geometric constraints from Section 3.2: a distance loss that keeps the inter-robot separation at the object length L_o, and a combined arm-and-face loss that keeps the follower oriented toward and facing the leader.

Detached Leader Gradients

Both formation losses are computed with the leader's predicted next state detached from the computational graph, so that gradients flow only into the follower's predicted action. This prevents the formation losses from pulling the leader off its goal-tracking trajectory. Instead, the follower learns to adapt to a leader that is free to focus on goal-tracking and

collision avoidance. We denote the detaching operation by the stop-gradient operator $\mathrm{sg}(\cdot)$.

Formation Distance Loss

The distance loss penalizes the absolute deviation of the predicted inter-robot distance from L_o. Let the predicted formation distance be

$$\hat{d}_m^{form} = \left\|\mathrm{sg}(\hat{p}_{m,1}) - \hat{p}_{m,2}\right\| \tag{4.22}$$

The distance loss is then

$$L_{form} = \frac{1}{|P|}\sum_{m\in P}\left|\hat{d}_m^{form} - L_o\right| \tag{4.23}$$

Arm and Face Loss

The arm-and-face loss combines two related orientation requirements that together specify the face-to-face grasping configuration. Although the problem formulation in Section 3.2 states only the heading-difference constraint, the arm component adds an implicit physical requirement: the follower must face toward the leader for the grasp to hold. This is not an extra constraint, but a consequence of the grasping assumption.

The arm component requires that the follower's heading $\hat{\theta}_{m,2}$ points toward the leader. Let ψ_m be the angle from the follower's predicted next position to the leader's predicted next position:

$$\psi_m = atan2(\mathrm{sg}(\hat{p}_{m,1,y}) - \hat{p}_{m,2,y}, \mathrm{sg}(\hat{p}_{m,1,x}) - \hat{p}_{m,2,x}) \tag{4.24}$$

The arm loss is active only when the angular gap between the follower's heading and ψ_m exceeds a tolerance ϵ_{arm}, which lets the follower's heading vary slightly without penalty:

$$L_{arm}^{base} = \frac{1}{|P|}\sum_{m\in P}\mathrm{ReLU}(\left|wrap(\psi_m - \hat{\theta}_{m,2})\right| - \epsilon_{arm}) \tag{4.25}$$

The face component enforces the two headings differ by π. Let the wrapped angular difference between the two predicted headings be:

$$\Delta\theta_m = \left|wrap(\mathrm{sg}(\hat{\theta}_{m,1}) - \hat{\theta}_{m,2})\right| \tag{4.26}$$

The face loss penalizes the deviation of this difference from π:

$$L_{face} = \frac{1}{|P|}\sum_{m\in P}|\Delta\theta_m - \pi| \tag{4.27}$$

The two components are combined with a fixed blending weight c_{face}:

$$L_{arm} = L_{arm}^{base} + c_{face}L_{face} \tag{4.28}$$

The arm term keeps the follower pointing roughly toward the leader, and the face term requires their headings to differ by π, enforcing the face-to-face grasping configuration.

4.3.5 Safety Loss

The safety loss provides a differentiable, soft-constraint approximation of the inter-group collision barrier. It reformulates the turning-circle barrier of the QP-CBF expert [1] as a differentiable loss, so that the network can be trained to satisfy inter-group safety without solving an optimization problem at each step. This embeds safety awareness directly into the learned policy.

Turning Center Construction

The barrier describes inter-group safety through the centers of the turning circles traced by each leader's predicted motion. For each leader of group m with predicted action $(v_{m,1}, \omega_{m,1})$, we compute a look-ahead position $\hat{p}_{m,1}^{look}$ at horizon T_{look} using the midpoint Euler rule:

$$\hat{p}_{m,1}^{look} = p_{m,1}(t) + v_{m,1}\begin{pmatrix}\cos(\theta_{m,1} + \frac{1}{2}\omega_{m,1}T_{look})\\ \sin(\theta_{m,1} + \frac{1}{2}\omega_{m,1}T_{look})\end{pmatrix}T_{look} \tag{4.29}$$

The turning center c_m is the center of the circular arc the leader would trace under its predicted action. It lies at distance r_{min} from $\hat{p}_{m,1}^{look}$, in the direction perpendicular to the leader's predicted heading:

$$c_m = \hat{p}_{m,1}^{look} + s_m r_{min}\begin{pmatrix}-\sin(\theta_{m,1} + \omega_{m,1}T_{look})\\ \cos(\theta_{m,1} + \omega_{m,1}T_{look})\end{pmatrix} \tag{4.30}$$

where $s_m = \text{sign}(\omega_{m,1})$ (with $s_m = 1$ when $\omega_{m,1} = 0$), selects the appropriate side of the arc.

Soft Barrier Loss

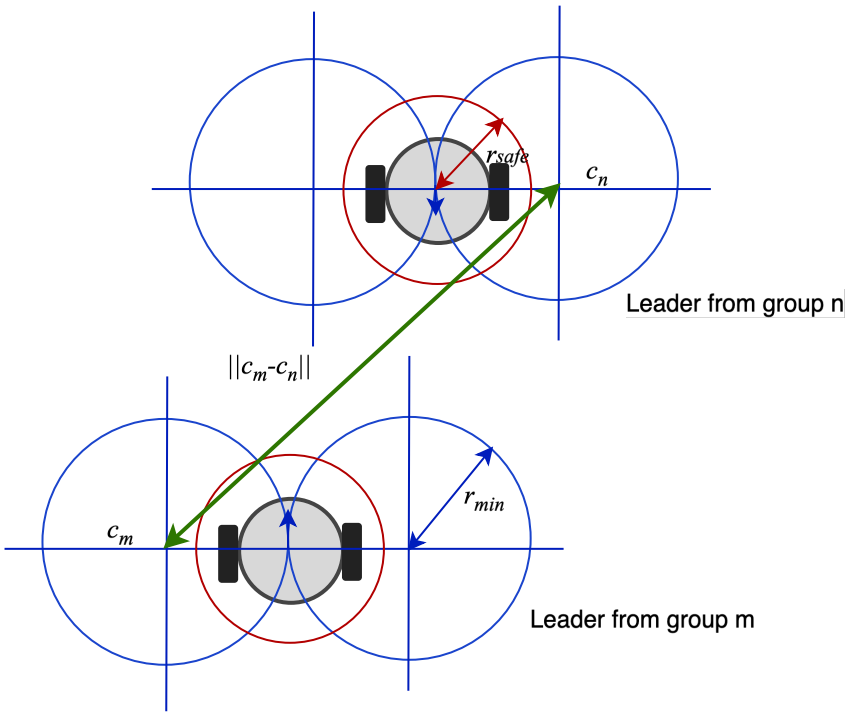


Figure 6. *Turning-circle construction for the inter-group safety loss.*

A unicycle turning under a nonzero angular velocity follows a circular arc. The center of this arc lies to one side of the robot, perpendicular to its heading, and either side is possible depending on the turning direction. The two blue circles in Figure 6 show these two possible turning circles, each of radius r_{min}. The sign $s_m = \text{sign}(\omega_{m,1})$ selects the side matching the leader's predicted angular velocity, so that c_m is the centre of the arc the leader follows.

The barrier requires the distance between the two centers $\|c_m - c_n\|$ to stay at least a safe distance apart:

$$\|c_m - c_n\| \geq d_{safe} = 2r_{safe} + 2r_{min} \tag{4.31}$$

Where r_{safe} is the safety radius around each robot. For each pair of leaders from different groups $m \neq n$, we define the barrier violation as

$$viol_{mn} = \text{ReLU}(d_{safe} - \|c_m - c_n\|) \tag{4.32}$$

Where $\text{ReLU}(\cdot)$ returns zero when the barrier is satisfied and grows linearly when it is violated. The safety loss takes the maximum squared violation over all valid leader pairs in the batch:

$$L_{safety} = \max_{(m,n)\in C} viol_{mn}{}^2, \quad C = \{(m,n): m \neq n, m, n \in P\} \tag{4.33}$$

We use the maximum not the mean so that the highest-risk pairs always produce a strong gradient, regardless of how many safe pairs are in the batch. This matters because inter-group conflicts are rare compared with intra-group interactions.

4.3.6 Composite Loss and Weight Allocation

All losses are combined into a single scalar training objective:

$L = \lambda_v L_v + \lambda_\omega L_\omega + \lambda_{goal} L_{goal} + \lambda_{stop} L_{stop} + \lambda_{head} L_{head} + \lambda_{form} L_{form} + \lambda_{arm} L_{arm} + \lambda_{safety} L_{safety}$.

The weights are fixed throughout training and were set by manual tuning to balance the relative magnitudes of the components. The BC weights λ_v and λ_ω serve as the base scale reference with equal weight for linear and angular velocity imitation. The physics-based weights are set much larger than the BC weights, because the main goal is for the network to learn the underlying behavior, not to memorize the expert's exact actions.

Among the physics-based weights, λ_{form} is the largest, since formation maintenance is a hard geometric constraint. λ_{safety} is large enough to penalize the worst-case inter-group violation. λ_{goal} is the dominant navigation signal. λ_{stop} and λ_{head} are set to similar values. λ_{arm} is the smallest, as it only refines the follower's orientation on top of the formation loss.

The BC terms still loosely keep the predicted actions within velocity ranges that the expert considers reasonable. This prevents the physics-based losses from pushing the network into dynamically unreasonable regions.

4.4 Training Procedure

The training dataset has two parts. The first part is an initial expert dataset. The second part is DAgger correction data, collected over several rounds.

4.4.1 Data Composition

The initial dataset is collected by running the QP-CBF expert on multi-group scenarios. These scenarios are designed to force frequent interactions between the different groups. In each scenario, the groups are placed on a collision course, and the meeting point is randomized near the center of the map. We keep only the scenarios in which all leaders reach their goals. If the QP-CBF solver becomes infeasible during an episode, that episode is discarded.

At each simulation step, we record one row of data for every robot. Each row stores the robot's position, its heading, the expert control action, the displacement from the robot to its goal, the distance between the two robots in the group, and the relative angle between their headings. These raw values are saved in Comma-Separated Values (CSV)

files. During training, the data loader reads these values and converts them into the normalized node features. The expert control action is used as the label for the behavioral cloning loss.

4.4.2 Dagger Aggregation Strategy

DAgger is applied for several rounds. The purpose is to extend the training data toward the states that the learned policy itself visits. The DAgger scenarios are harder than the initial ones. The number of groups in each episode is randomized between two and six. The robots start from the field boundary, and their goals are placed on the far side, so that they must travel across the whole field and interact with many other groups.

In each round, the learner policy and the QP-CBF expert are mixed to drive the simulation, controlled by a coefficient β. At each step, the executed action is taken from the expert with probability β, and from the learner $(1-\beta)$. The expert is queried at every step to provide the corrective label, even when the learner action is executed. Following the DAgger schedule, β starts large in the early rounds, so the expert guides most of the motion. It is then reduced over the rounds and reaches zero in the final round. At that point the learner drives the simulation entirely on its own, and the collected states fully reflect the learner's own behavior. If the solver crashes because of a deadlock, that episode is discarded. Timeout episodes are kept, because the states visited during these failed attempts are the out-of-distribution states that the network needs to learn from.

The aggregation follows an offline strategy, so that each round is collected as a complete dataset, stored separately, and merged with all previous rounds before the next training run. This offline strategy decouples data collection from gradient updates and ensures the model is always trained on the union of all data collected so far.

5 SIMULATION & DISCUSSION

This chapter presents the simulation study used to evaluate the proposed GNN-based control policy against the QP-CBF expert baseline. The evaluation is organized around the three research questions stated before. The first question is whether the GNN can learn the formation-keeping and goal-tracking behavior required for single-group transport. The second question is whether the soft-constraint formulation resolves the deadlock and scalability limitations of the QP-CBF in multi-group scenarios, and at what cost. The third question is whether the learned policy generalizes to group densities beyond those seen during training.

Section 5.1 specifies the simulation parameters and the evaluation metrics. Sections 5.2 to 5.3 present the two experimental scenarios in order of increasing complexity. Each scenario is followed by an analysis of its results. Section 5.4 closes the chapter with a cross-scenario discussion.

5.1 Simulation Setup

5.1.1 Environment and Parameters

All experiments are conducted in a Python-based simulation environment, using the unicycle kinematic model described in Section 3.1.

The physical and environment parameters shared across all scenarios are summarized in Table 5.1. The maximum linear and angular velocities follow the actuator limits defined in Section 3.2. The safety radius r_{safe} and the minimum turning radius r_{min} are taken directly from the QP-CBF expert formulation, so that the two methods can be compared fairly. The communication radius r_{comm} is the distance threshold below which two leaders form an inter-group edge in the graph. Scenario-specific parameters, such as the target formation distance L_o, the success criteria, and the episode duration, are defined at the beginning of each scenario section.

Table 5.1. *Physical & Environment Parameters*

Parameter	Symbol	Value
Max linear velocity	V_{max}	2 m/s
Max angular velocity	Ω_{max}	10 rad/s
Time step	Δt	0.05 s
Speed decay	k_v	0.1
Parking tolerance	ϵ_{park}	0.1 m
Body safety radius	r_{safe}	0.2 m
Min turning radius	r_{min}	0.3 m
Communication radius	r_{comm}	5.0 m

The GNN model uses the architecture and training configuration summarized in Table 5.2. These hyperparameters are fixed across all scenarios and correspond to the final model described in Chapter 4.

Table 5.2. *GNN Architecture & Training Hyperparameters*

Parameter	Value
Node feature dimension	10
Hidden dimension d_h	128
GAT layers	3
Attention heads (L1, L2, L3)	$4, 4, 1$
Optimizer	Adam
Learning rate	1.5×10^{-4}
Batch size	8192
Training epochs	800
Gradient clip norm	1.0

The loss weights used during training are listed in Table 5.3. As discussed, the physics-based terms carry much larger weights than the behavioral cloning terms. This encourages the network to learn the underlying behavior instead of imitating the expert's exact actions.

Table 5.3. *Loss Weight Values*

Component	Symbol	Value
Linear velocity BC	λ_v	1.0
Angular velocity BC	λ_ω	1.0
Goal distance	λ_{goal}	30.0
Stop	λ_{stop}	40.0
Heading	λ_{head}	40.0
Formation distance	λ_{form}	60.0
Arm and face	λ_{arm}	30.0
Safety	λ_{safety}	50.0

5.1.2 Baselines

The QP-CBF expert controller[1] serves as the primary reference throughout the evaluation. The expert solves a quadratic program at each control step to compute an action that satisfies the hard formation and safety constraints. It represents the best performance achievable under the optimization-based paradigm. In scenarios where it works, it provides an upper bound on formation precision and goal-tracking accuracy.

5.1.3 Evaluation metrics

Seven metrics are used to evaluate each method. As summarized in Table 5.4, the first four apply to all scenarios and the last three are specific to multi-group settings.

Table 5.4. *Evaluation Metrics*

Metric	Notation	Scenarios
Success Rate	SR	All
Average Navigation Time	$\bar{T}$	All
Mean Formation Distance Error	$\bar{e}_{form}$	All
Mean Face Angle Error	$\bar{e}_{face}$	All
Minimum Inter-group Distance	d_{min}	Multi-group
Safe Ratio	SR_{safe}	Multi-group
Deadlock Rate	DR	Multi-group (expert only)

Success Rate (SR) is the fraction of episodes in which all groups satisfy their success conditions within the time limit. The conditions differ by scenario and are defined in each scenario section.

Average Navigation Time ($\bar{T}$) is the mean time to completion across successful episodes. Episodes that fail or time out are excluded from this metric.

Mean Formation Distance Error ($\bar{e}_{form}$) is the time-averaged absolute deviation of the inter-robot distance from the target object length L_o, computed across all time steps and all groups within each episode:

$$\bar{e}_{form} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N}\sum_{m=1}^{N}\left|\left\|p_{m,1}(t) - p_{m,2}(t)\right\| - L_o\right| \tag{5.1}$$

Mean Face Angle Error ($\bar{e}_{face}$) is the time-averaged deviation of the inter-robot heading difference from the ideal face-to-face configuration of π:

$$\bar{e}_{face} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N}\sum_{m=1}^{N}\left|\left|wrap(\theta_{m,1}(t) - \theta_{m,2}(t))\right| - \pi\right| \tag{5.2}$$

Minimum Inter-group Distance (d_{min}) is the smallest distance observed between leaders of different groups over the full episode. It measures how closely the groups approach one another during navigation.

Safe Ratio (SR_{safe}) is the fraction of successful episodes in which the minimum inter-group distance remains above $2r_{safe}$, no physical collision occurs.

$$SR_{safe} = \frac{number\ of\ episodes\ where\ d_{min} \geq 2r_{safe}}{number\ of\ successful\ episodes} \tag{5.3}$$

Deadlock Rate (DR) is the fraction of episodes in which the expert's QP-CBF solver becomes infeasible and fails to return a valid control input, ending the episode before the time limit:

$$DR = \frac{number\ of\ deadlock\ episodes}{total\ episodes} \tag{5.4}$$

A deadlock happens when the inter-group safety constraints and the intra-group formation constraints conflict at the same time, leaving no feasible solution to the quadratic

program. This metric applies only to the QP-CBF expert, since the GNN policy does not solve an optimization problem and therefore cannot deadlock.

5.2 Scenario 1: Single-Group Collaborative Transport

This scenario evaluates the basic capability of the GNN to learn formation-keeping and goal-tracking for a single leader-follower group. It also examines how well the policy generalizes across a range of object lengths that the training distribution does not fully cover. The scenario parameters are listed in Table 5.5.

Table 5.5. *Single-Group Scenario Parameters*

Parameter	Value
Target formation distance L_o	0.3- 2.0 m (step 0.1 m)
Goal reaching tolerance ϵ_{goal}	0.05 m
Formation success tolerance ϵ_{form}	0.05 m
Face-to-face tolerance ϵ_{face}	0.5 rad
Max episode duration	20 s
Trials per L_o	50

The training dataset for this scenario covers discrete object lengths of 0.3, 0.5, 0.7, 1.0, 1.2 and 1.5 m. The data is not evenly distributed across these lengths. Most of the data lies in the shorter range, from 0.3 to 1.0 m, while the longer lengths 1.2 and 1.5 m have far fewer samples. This choice lets the network first learn the basic formation-and-transport pattern on stable short-object cases. Mixing in many long lengths early would change L_o too often and make it harder for the network to learn a consistent pattern.

Values between these training points are unseen interpolation targets, and values beyond 1.5 m up to 2.0 m, are extrapolation targets. An episode is considered successful if the leader reaches within $\epsilon_{goal} = 0.05$ m of the goal while the formation distance is within ϵ_{form} of L_o at the same time step. The face-to-face configuration is not part of the success condition. We still report the mean face angle error $\bar{e}_{face}$ to measure how well each method holds the face-to-face tolerance ϵ_{face}.

5.2.1 Results

Table 5.6 (a) and Table 5.6 (b) report the evaluation results for the GNN and the QP-CBF expert at representative object lengths. The full trends across all tested lengths $L_o \in \{0.3, 0.4, \ldots, 2.0\}$ are shown in Figure 7.

Table 5.6 (a). *Success Rate and Navigation Time*

L_o (m)	SR GNN	SR Expert	$\overline{T}$ GNN (s)	$\overline{T}$ Expert (s)
0.3	100%	100%	5.54	4.57
0.5	100%	100%	5.10	4.69
0.7	100%	100%	5.30	4.79
1.0	76%	100%	5.32	4.88
1.2	72%	100%	5.75	4.82
1.5	64%	100%	5.42	4.97
1.7	30%	100%	5.27	5.46
2.0	54%	100%	7.84	5.60

Table 5.6 (b). *Formation Distance Error and Face Angle Error*

L_o (m)	$\overline{e}_{form}$ GNN(m)	$\overline{e}_{form}$ Expert(m)	$\overline{e}_{face}$ GNN (rad)	$\overline{e}_{face}$ Expert (rad)
0.3	0.034	0.002	0.143	0.054
0.5	0.019	0.002	0.137	0.098
0.7	0.020	0.004	0.159	0.136
1.0	0.017	0.005	0.305	0.178
1.2	0.021	0.006	0.369	0.202
1.5	0.029	0.008	0.474	0.265
1.7	0.028	0.008	0.686	0.324
2.0	0.057	0.015	0.682	0.387

For short objects up to $L_o \leq 0.9$ m, the GNN achieves 100% success rate, comparable navigation time to the expert, and formation errors below 0.035 m. From $L_o = 1.0$ m onward, the success rate drops. It falls to 76% at $L_o = 1.0$ m and continues down to 30% at $L_o = 1.7$ m. The face angle error rises with L_o and passes the tolerance $\epsilon_{face} = 0.5$ rad, reaching 0.68 rad at $L_o = 2.0$ m. The expert keeps a 100% success rate across all lengths, with consistently lower formation and face angle errors.

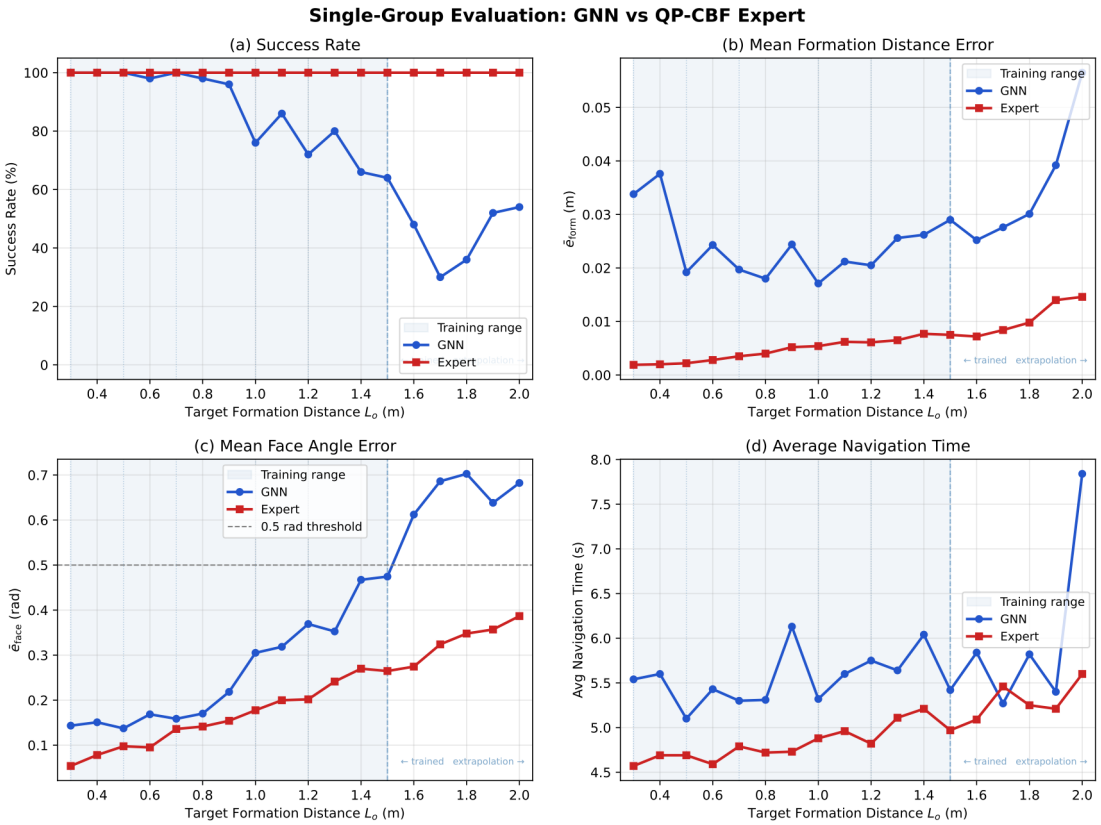


Figure 7. *Single-Group Results Comparison: GNN vs QP-CBF Expert.*

Figure 7 shows the full trends across all tested lengths. In plot (a), the GNN success rate stays near 100% for short objects and starts to drop after $L_o = 1.0$ m, reaching a low of 30% at $L_o = 1.7$ m. The expert stays at 100% throughout. In plot (b), both methods show increasing formation error as L_o grows, and the GNN's error is higher than that of the expert at every length. In plot (c), the GNN face angle error crosses 0.5 rad between $L_o = 1.5$ m and $L_o = 1.7$ m, and keeps rising to 0.68 rad, while the expert stays below 0.39 rad. In plot (d), navigation times are similar for short objects, but the GNN spikes to 7.84 s at $L_o = 2.0$ m.

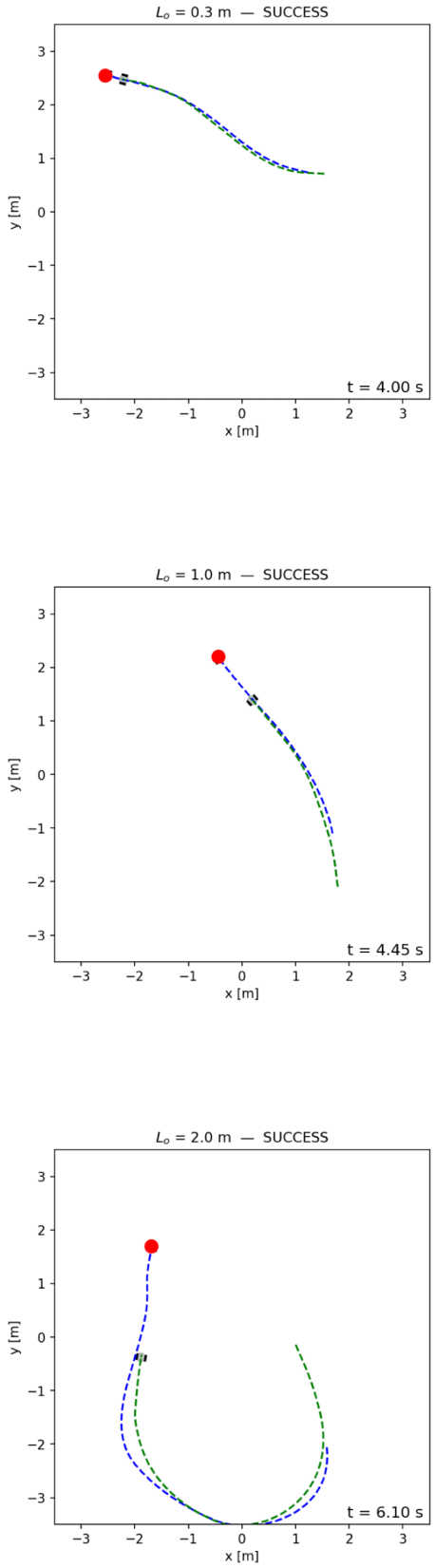
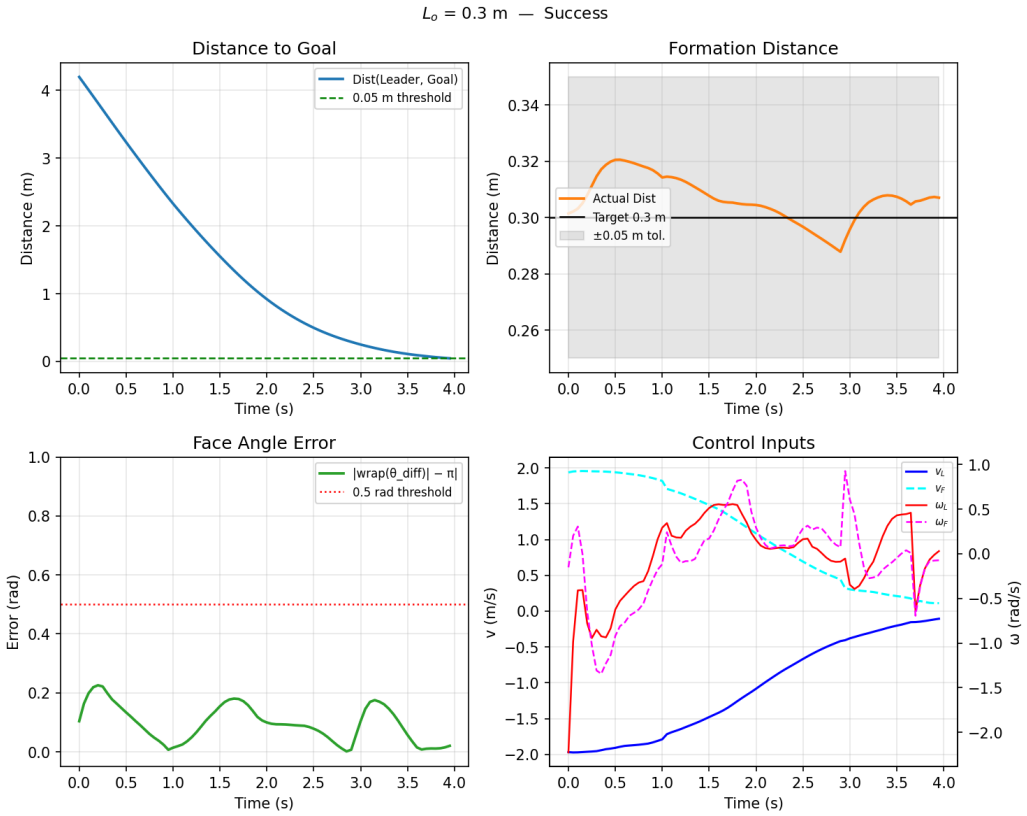
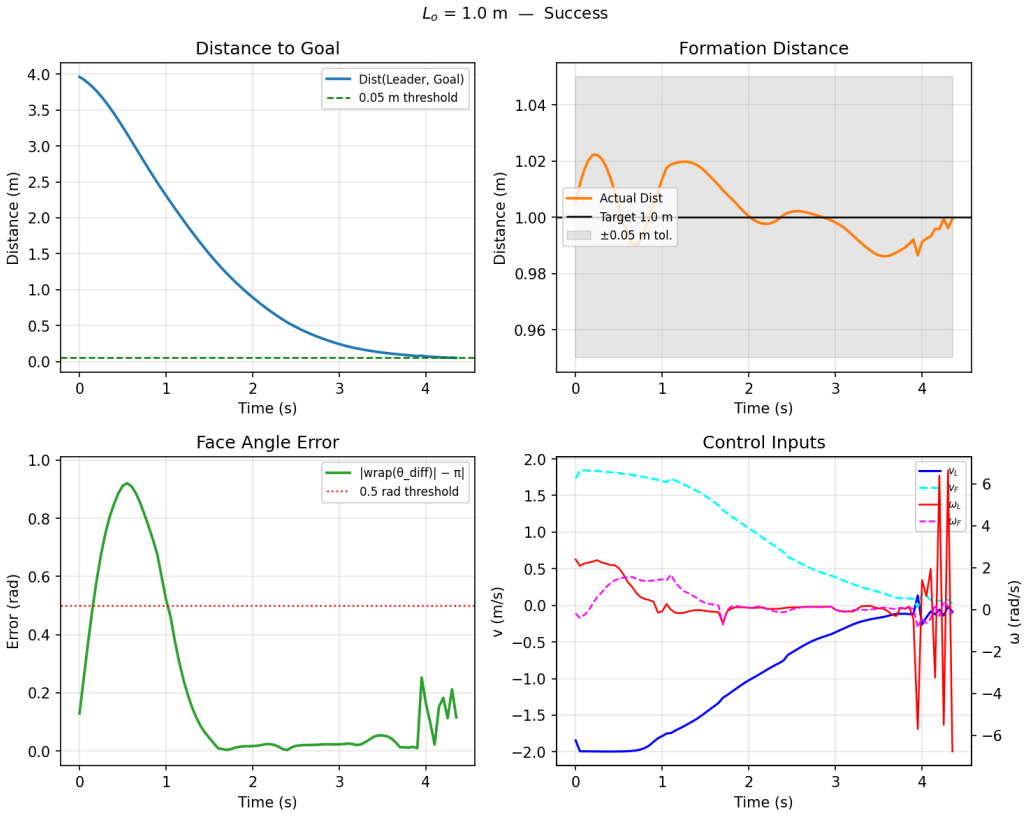


Figure 8. *Trajectories of the GNN policy for Single-Group with* $L_o = 0.3, 1.0, 2.0$ *m.*

Figure 8 shows the trajectories for the three representative episodes. At $L_o = 0.3$ m, the two paths overlap almost completely, since the robots are close together. At $L_o = 1.0$ m,

a small gap appears toward the end, as the robots turn into the goal. At $L_o = 2.0$ m, the two paths are separated, and the follower traces a wider outer arc. This episode also takes longer, 6.10 s compared with 4.00 s at $L_o = 0.3$ m.





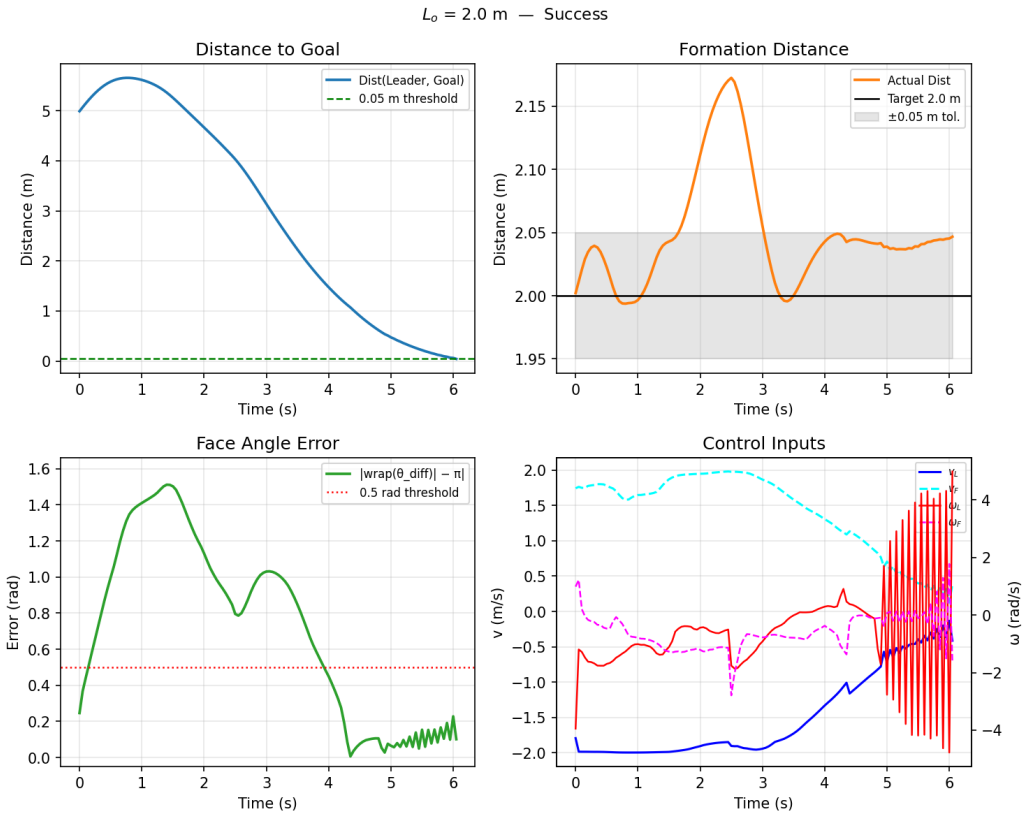


Figure 9. *Time histories of the GNN policy for Single-Group with* $L_o = 0.3, 1.0, 2.0$ *m.*

Figure 9 shows the corresponding metrics over time. At $L_o = 0.3$ m, the distance to goal decreases smoothly, formation distance stays within tolerance, and face angle error stays well below 0.5 rad throughout. At $L_o = 1.0$ m, the distance to the goal again decreases smoothly, but the face angle error rises above 0.5 rad early in the episode, peaking near 0.9 rad, before settling close to zero for the rest of the run. At $L_o = 2.0$ m, the distance to goal first increases before the robots turn toward the target. Formation distance rises to about 2.17 m before recovering to within tolerance. Face angle error stays above 0.5 rad for most of the first half, peaks at about 1.5 rad, and only drops below the threshold after $t = 4$ s. Large angular velocity spikes appear in the final part of the episode, as the network corrects heading to reach the goal.

5.2.2 Analysis

For short objects up to $L_o \leq 0.7$ m, the GNN achieves a high success rate, with formation errors below 0.035 m and face errors below 0.16 rad. This confirms that the GNN can learn the formation-keeping and goal-tracking behavior needed for single-group collaborative transport. The training data only covers discrete lengths 0.3, 0.5, 0.7, 1.0, 1.2 and 1.5 m, but the GNN also performs well at untrained values between these points. This

shows that the network has learned a continuous representation of the formation geometry, instead of only memorizing specific training points.

Performance degrades from $L_o = 1.0$ m, and success rate drops to between 30% and 54% beyond $L_o = 1.5$ m. This is partly due to the training data distribution. The dataset is concentrated at shorter lengths, with most episodes below $L_o = 1.0$ m, fewer episodes at 1.2 m and 1.5 m, and no data beyond 1.5 m. Long-object transport is also a harder problem in general, as we can see even with hard constraints, its formation error grows from 0.002 m to 0.015 m and its face error from 0.054 rad to 0.387 rad across the tested range. When the inter-robot distance is large, any rotation requires the outer robot to travel a wider arc at higher speed, which makes coordination harder for both methods.

It is worth noting that the GNN still reaches 30 to 54% success at $L_o \geq 1.5$ m, a range almost absent from the training data. The network has seen very few or no examples of objects longer than 1.5 m, yet it can still complete the task in a portion of cases by generalizing the formation patterns learned at shorter lengths. This suggests that the formation patterns the network learned at shorter lengths extend to object lengths it never saw.

Across all lengths, the expert achieves 100% success with formation errors roughly an order of magnitude lower than the GNN. This gap comes from the difference between the two approaches. The expert enforces formation as a hard constraint through exact QP optimization, so the constraint is satisfied at every step. The GNN treats formation as a soft objective in the composite loss, so it can only approximate the constraint. This is a general property of learning-based soft-constraint formulations. However, this soft formulation also gives the GNN flexibility in the multi-group scenarios.

Although the expert achieves higher accuracy across all lengths, it solves a QP optimization at every control step, and its computation time per step grows with L_o as the constraint coupling becomes tighter. The GNN runs a single forward pass, with a constant inference time that does not depend on L_o. This computational difference is small in the single-group setting, but it becomes significant in the multi-group scenarios examined next.

5.3 Scenario 2: Multi-Group Collaborative Transport

This scenario evaluates the GNN's ability to handle multi-group coordination. Several robot groups must navigate toward their respective goals while avoiding collisions with each other. The scenario parameters are listed in Table 5.7.

Table 5.7. *Multi-Group Scenario Parameters*

Parameter	Value
Number of groups N	2, 3, 4, 5, 6, 7, 8, 9, 10
Target formation distance L_o	0.3 m
Goal reaching tolerance ϵ_{goal}	0.2 m
Safety collision threshold	$2r_{safe} = 0.4$ m
Max episode duration	100 s
Trials per N	50

The model is trained on scenarios with two to six groups. The training data is not evenly distributed across these counts. Most of the data comes from the two-group to four-group cases. The five-group and six-group cases are included, but with fewer samples. This is because the expert is slow to run with many groups, and it also fails more often in dense scenarios, where the QP solver deadlocks. Both effects make expert data harder to collect as the number of groups grows. The group counts from seven to ten are absent from the training data, so they test how well the policy generalizes to densities beyond the training range.

An episode is considered successful if all leaders reach within $\epsilon_{goal} = 0.2$ m of their goals within the time limit. The tolerance is wider compared to the single-group scenario, because multi-group navigation involves frequent detours to avoid other groups, which makes precise goal reaching harder. Formation quality is not part of the success condition, but it is reported separately through $\bar{e}_{form}$ and $\bar{e}_{face}$.

5.3.1 Results

Table 5.8(a) and Table 5.8(b) report the evaluation results for the GNN and the QP-CBF expert across group counts $N \in \{2,3,\dots,10\}$. The full trends are shown in Figure 10.

Table 5.8 (a). *Success Rate, Deadlock Rate, and Safe Ratio*

N	SR GNN	SR Expert	DR Expert	SR_{safe} GNN	SR_{safe} Expert
2	100%	100%	0%	94%	100%
3	100%	86%	12%	84%	100%
4	100%	80%	18%	74%	93%
5	100%	64%	16%	84%	75%
6	100%	36%	24%	82%	56%
7	90%	18%	40%	89%	67%
8	98%	4%	38%	84%	50%
9	82%	4%	30%	73%	50%
10	92%	4%	40%	78%	0%

Table 5.8 (b). *Formation Distance Error and Face Angle Error*

N	$\bar{e}_{form}$ GNN(m)	$\bar{e}_{form}$ Expert(m)	$\bar{e}_{face}$ GNN (rad)	$\bar{e}_{face}$ Expert (rad)
2	0.130	0.002	0.075	0.013
3	0.209	0.006	0.107	0.041
4	0.160	0.010	0.103	0.069
5	0.200	0.015	0.131	0.126
6	0.234	0.019	0.155	0.187
7	0.434	0.020	0.228	0.201
8	0.303	0.019	0.204	0.202
9	0.466	0.021	0.266	0.215
10	0.431	0.021	0.265	0.194

For $N \leq 6$, the GNN keeps a 100% success rate, while the expert's success rate drops from 100% at $N = 2$ to 36% at $N = 6$. Beyond the training range, the GNN stays at 82% or higher through $N = 10$, while the expert falls to 4%. The expert's deadlock rate rises with the number of groups, reaching 40% at $N = 7$ and $N = 10$. The GNN's formation and face errors are higher than the expert's at every group count. However, its safe ratio overtakes the expert from $N = 5$ onward, and the gap widens as the scene becomes denser.

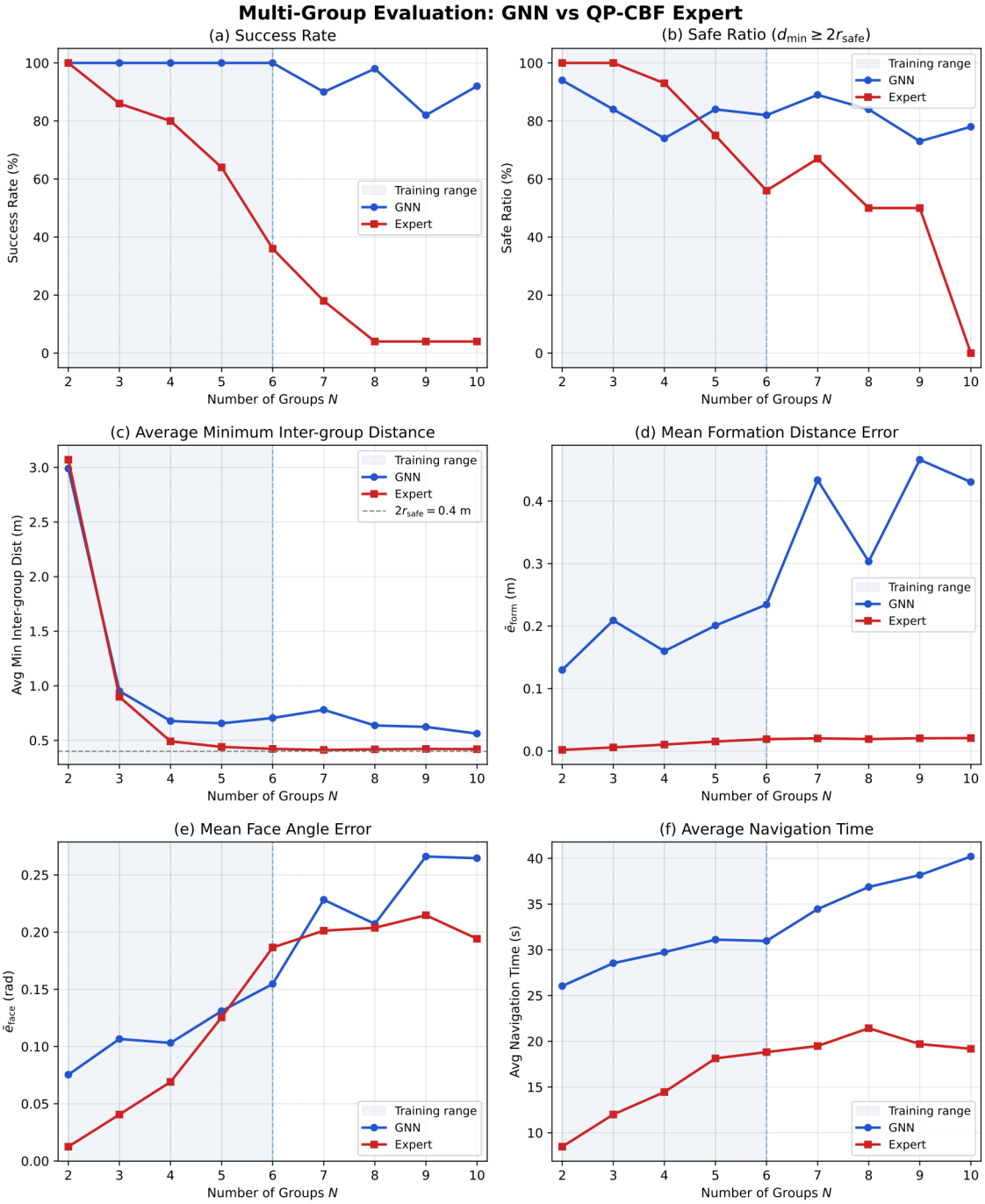


Figure 10. *Multi-Group Results Comparison: GNN vs QP-CBF Expert.*

Figure 10 compares the GNN and the expert as the number of groups N increases from 2 to 10. The shaded region marks the training range of 2 to 6 groups, and the counts from 7 to 10 test generalization beyond it. The dashed line in plot (c) marks the collision distance $2r_{safe} = 0.4$.

In plot (a), the GNN keeps a 100% success rate up to six groups and stays at 82% or higher through ten groups. The expert starts at 100% but drops steadily, reaching 36% at six groups and 4% at eight groups and beyond. In plot (b), the expert has a higher

safe ratio at two and three groups, but the GNN overtakes it from five groups onward. At ten groups, the GNN stays near 78%, while the expert drops to 0%. In plot (c), the expert's average minimum inter-group distance sits close to the 0.4 m collision line for four or more groups, while the GNN keeps a larger distance at every count. In plot (d) and (e), the expert keeps very low formation and face errors, while the GNN's errors are higher and grow with the number of groups. In plot (f), the GNN takes longer to finish than the expert at every group count, and its time rises to about 40 s at ten groups.

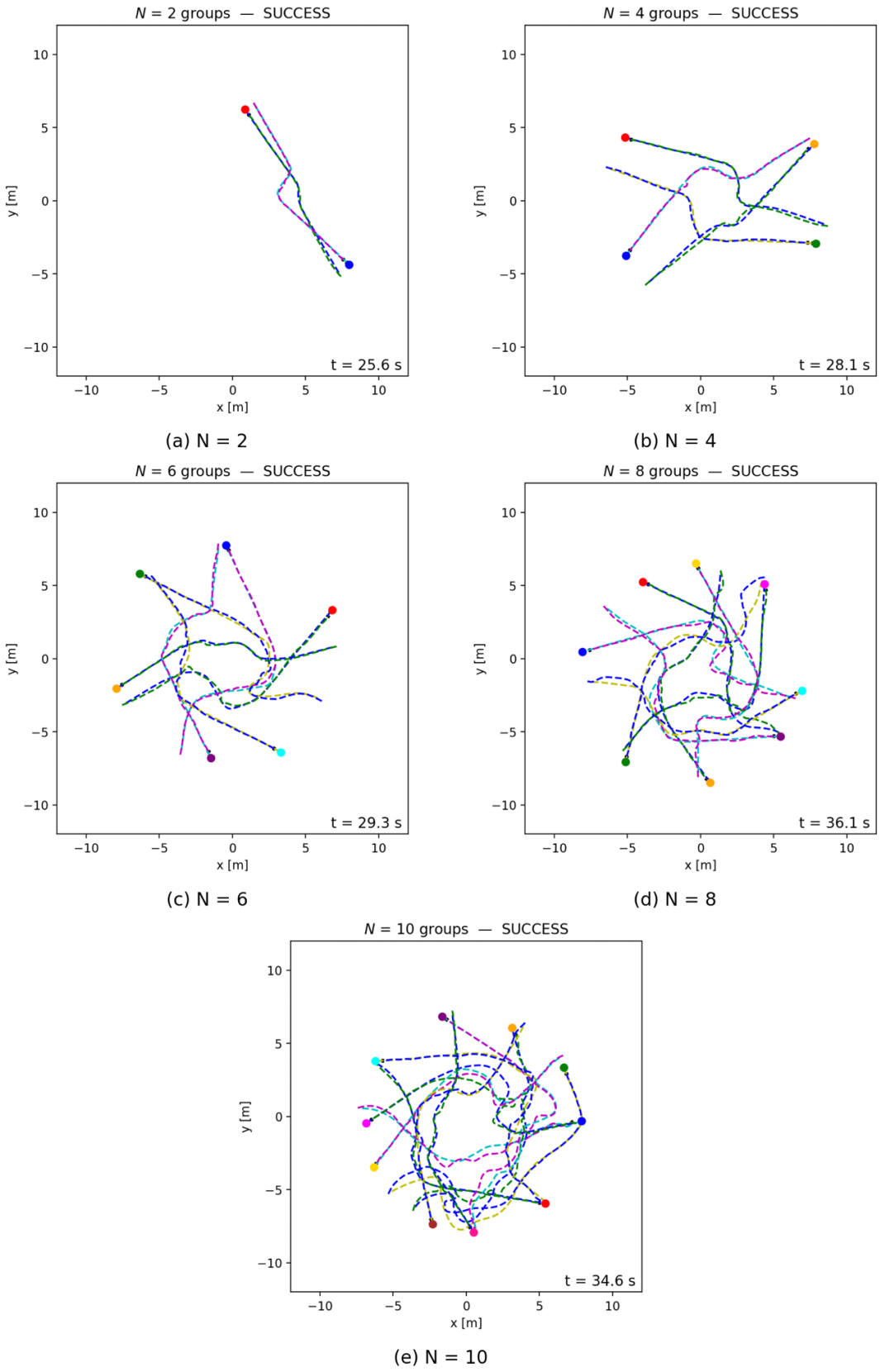


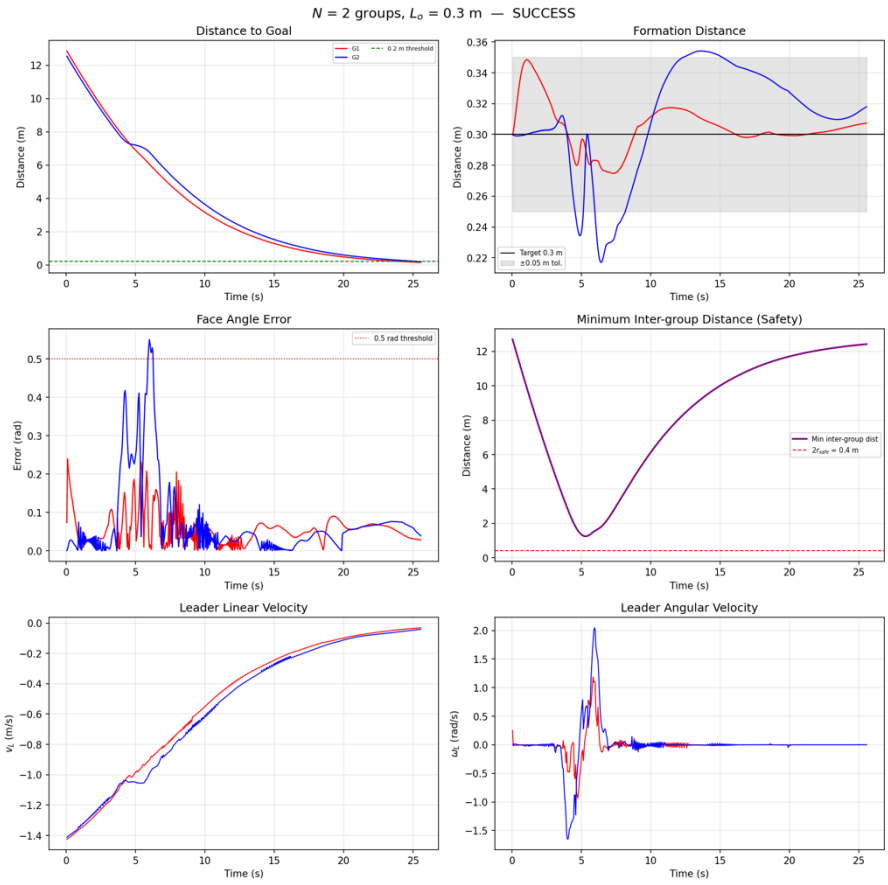
Figure 11. *Trajectories of the GNN policy for 2, 4, 6, 8, and 10 groups.*

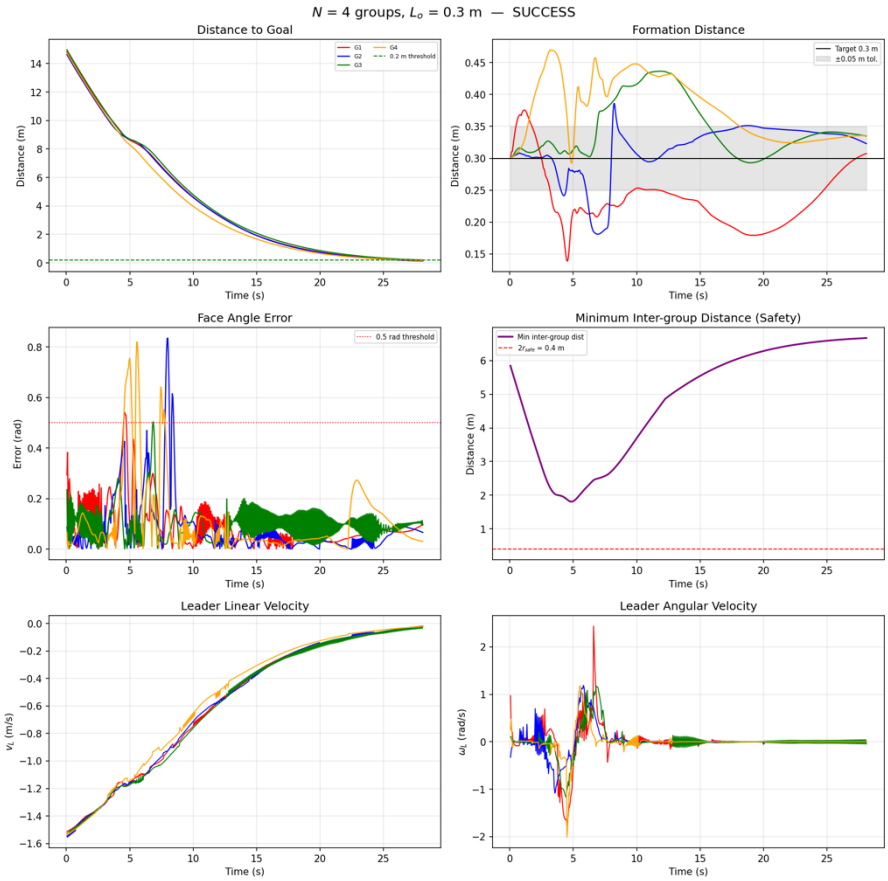
Figure 11 shows trajectories of the GNN policy for 2, 4, 6, 8, and 10 groups. We show these five groups to cover the range from sparse to dense scenes. The skipped counts

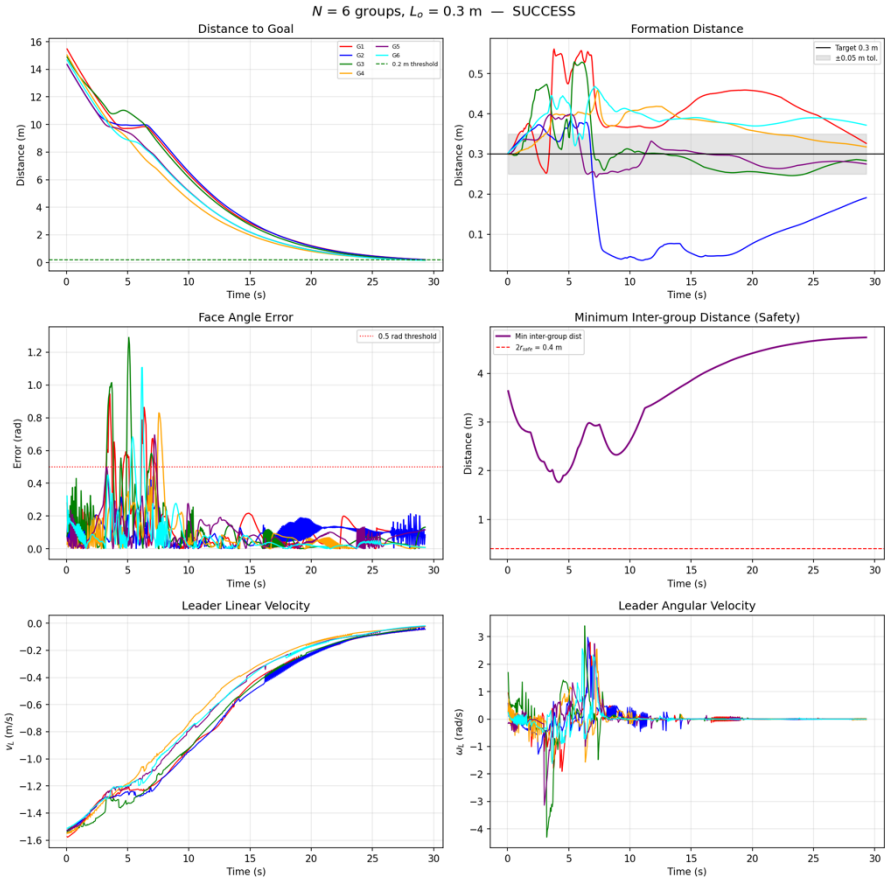
behave like their neighbors, so adding them would repeat the same pattern. Each filled circle marks a goal, and each dashed curve is the path of one robot. All five episodes end in success, with every leader reaching its goal. The model is trained on scenarios with up to six groups. The runs at eight and ten groups are beyond this training range.

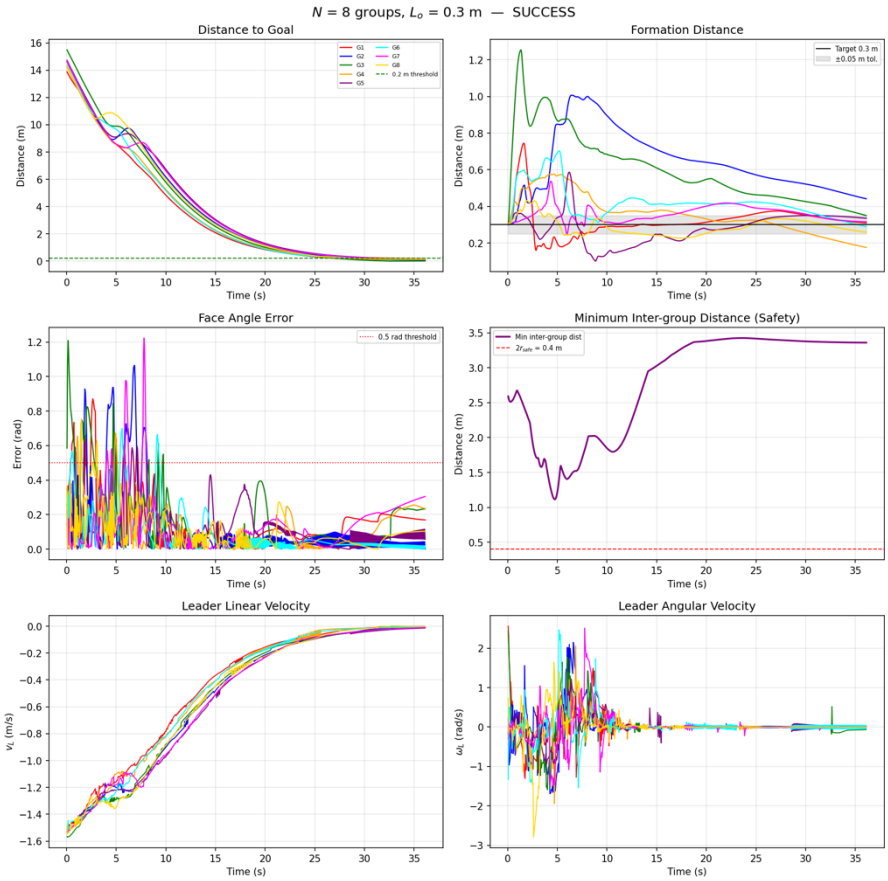
In every scenario, the start and goal of each group are placed on opposite sides of the map. Each group must therefore cross the map and pass through the others on the way. With two groups, the two paths meet once. Both paths bend to the side to keep a safe distance, then straighten again after the groups pass. This is the avoidance the GNN has learned. When two groups come close, each one turns away from the other.

With more groups, these crossings happen more often and overlap. Each group turns away when it comes near another, but it now has to do so several times before it reaches its goal. This produces the curved, looping paths seen at six, eight, and ten groups. No path is blocked, and all groups still reach their goals. The cost is a longer route. The navigation time grows from 25.6 s at two groups to 34.6 s at ten groups.




N = 4 groups, L_o = 0.3 m — SUCCESS
Distance to Goal
Distance (m)
Time (s)
G1
G2
G3
G4
0.2 m threshold
Formation Distance
Target 0.3 m
±0.05 m tol.
Face Angle Error
Error (rad)
0.5 rad threshold
Minimum Inter-group Distance (Safety)
Min inter-group dist
2r_safe = 0.4 m
Leader Linear Velocity
v_L (m/s)
Leader Angular Velocity
ω_L (rad/s)


N = 6 groups, L_o = 0.3 m — SUCCESS
Distance to Goal
Distance (m)
Time (s)
G1
G2
G3
G4
G5
G6
0.2 m threshold
Formation Distance
Target 0.3 m
±0.05 m tol.
Face Angle Error
Error (rad)
0.5 rad threshold
Minimum Inter-group Distance (Safety)
Min inter-group dist
2r_safe = 0.4 m
Leader Linear Velocity
v_L (m/s)
Leader Angular Velocity
ω_L (rad/s)


N = 8 groups, L_o = 0.3 m — SUCCESS
Distance to Goal
Distance (m)
Time (s)
0.2 m threshold
Formation Distance
Target 0.3 m
±0.05 m tol.
Face Angle Error
Error (rad)
0.5 rad threshold
Minimum Inter-group Distance (Safety)
Min inter-group dist
2r_safe = 0.4 m
Leader Linear Velocity
v_L (m/s)
Leader Angular Velocity
ω_L (rad/s)

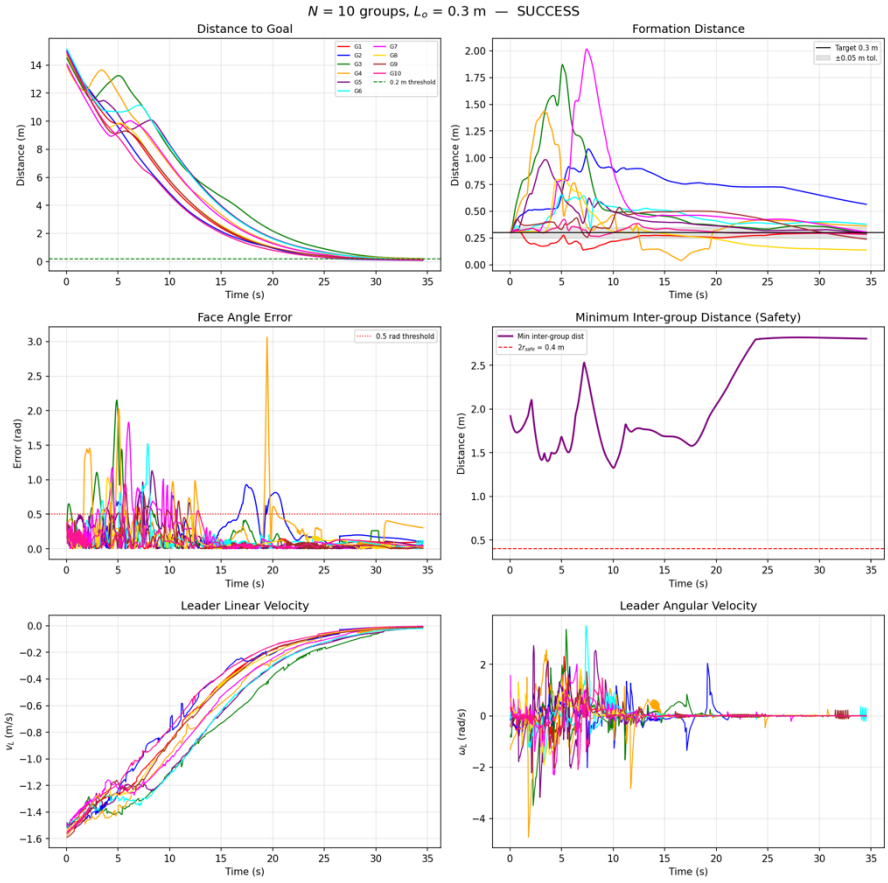

N = 10 groups, L_o = 0.3 m — SUCCESS
Distance to Goal
Distance (m)
Time (s)
0.2 m threshold
Formation Distance
Target 0.3 m
±0.05 m tol.
Face Angle Error
Error (rad)
0.5 rad threshold
Minimum Inter-group Distance (Safety)
Min inter-group dist
2r_safe = 0.4 m
Leader Linear Velocity
v_L (m/s)
Leader Angular Velocity
ω_L (rad/s)

Figure 12. *Time histories of the GNN policy for 2, 4, 6, 8, and 10 groups.*

Figures 12 shows the time histories of the same five episodes, for 2, 4, 6, 8, and 10 groups. Each figure has six plots: distance to goal, formation distance, face angle error, minimum inter-group distance, leader linear velocity, and leader angular velocity.

The distance-to-goal curves have the same shape in all five cases. Every group starts about 14 to 15 m from its goal, and the distance falls to below the 0.2 m threshold by the end. The leader linear velocity also follows the same pattern. It starts near the maximum of 1.6 m/s and decays toward zero as the groups reach their goals. The sign is negative because the leader travels backward.

The differences appear while the groups meet and avoid each other. In the formation distance plots, the curves stay near the 0.3 m target at the start and end, but they deviate while the groups are close. The deviation grows with the number of groups. With two groups, the formation distance stays within about 0.22 to 0.35 m. With ten groups, some groups reach 2.0 m before returning to the target. The face angle error shows the same trend. Its peaks grow from about 0.55 rad at two groups to over 3 rads at ten groups. The leader angular velocity has its largest swings during the same period, which matches the turning needed to avoid other groups.

The minimum inter-group distance stays above the $2r_{safe} = 0.4$ m line in all five episodes. Each curve dips while the groups are close, reaches its lowest point when they are nearest, and rises again as they separate. Across the five episodes, it stays between about 1.1 and 1.8 m. In every case the distance stays well above the collision line.

5.3.2 Analysis

The multi-group results test whether the soft-constraint formulation can handle the deadlock and scalability limits of the QP-CBF expert, and at what cost. The expert's success rate falls as the number of groups grows. Its failures take two forms. One is the deadlock, where the expert reports that no feasible control exists and the episode stops. The other is the timeout, where the solver still returns a control, but the groups stop moving and never reach their goals. The deadlock rate is listed in Table 5.8(a). The timeout rate is not, but it equals $100\% - SR - DR$. From five groups on, timeouts make up more than half of all expert failures, and about two thirds at nine groups. The expert fails mostly by stopping, not by reporting an error.

Both forms have the same cause. The expert must satisfy several hard constraints at once: the formation distance and heading inside each group, the safety distance to every other group, and progress toward each goal. When the groups are spread out these do

not conflict. When many groups enter the same region, no single control satisfies all of them, so the solver either reports infeasibility or returns a control that holds the groups in place. Figure 13 shows both cases.

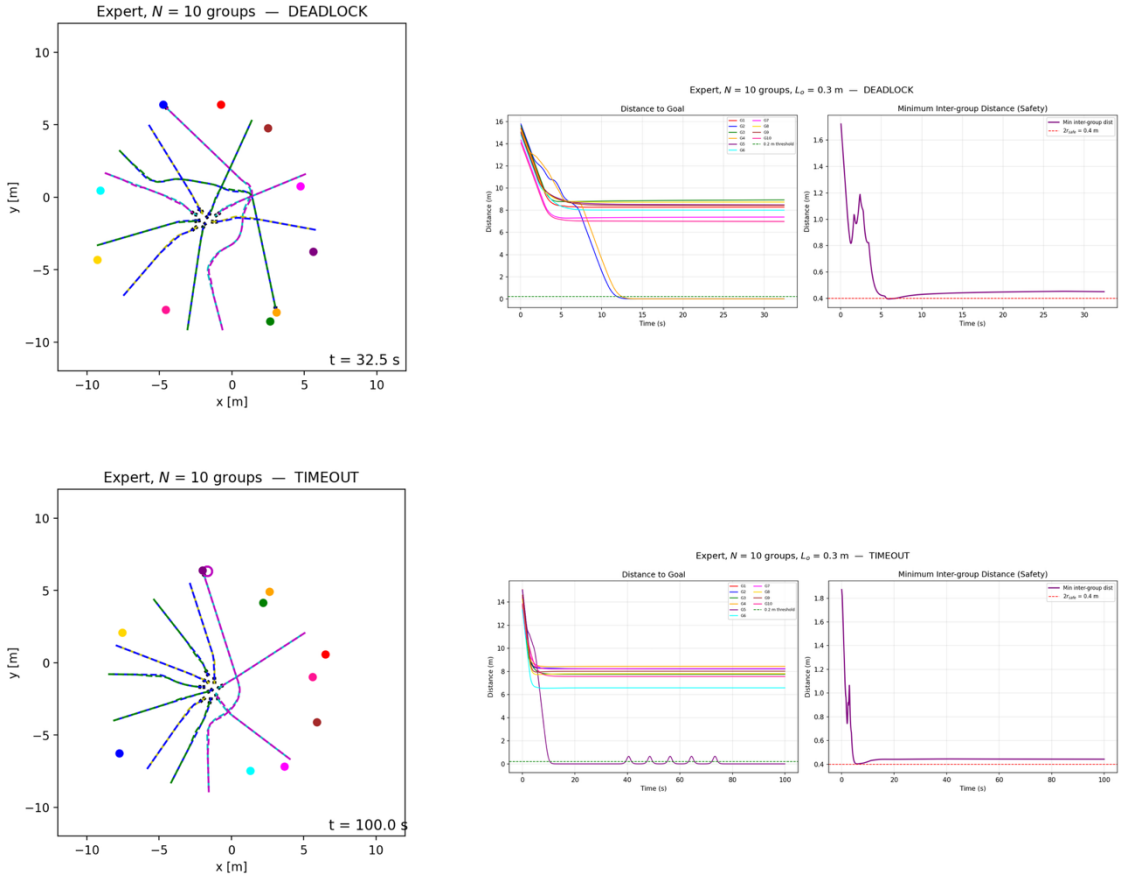


Figure 13. *Expert's performance in 10 groups.*

The GNN does not have this problem. Its success rate stays at 82% or above through ten groups, including the seven-to-ten range that is absent from the training data. Since inter-group formation is not part of the success test, this rate reports whether every group reaches its goal, which it does in most runs. The expert's groups drive toward the center and stop once the QP can no longer find a safe control in Figure 13. The GNN's groups turn while the others are still far away, so the paths form a ring around the center not meeting at one point in Figure 11. The GNN never solves for a control that satisfies every constraint at once, so the situation that jams the expert does not arise. This early avoidance is the behavior the weighted loss terms produced in training, and the soft formulation removes the deadlock and lets the method scale to denser scenes.

The cost appears in accuracy. The expert holds the formation almost exactly, with a formation distance error between 0.002 and 0.021 m and a face error that stays under the 0.5 rad tolerance. The GNN's formation errors are larger and grow with the count, because its groups turn aside and take longer paths to avoid one another. The expert

keeps the formation tight because the QP enforces it as a hard constraint. The GNN treats the formation as one soft term among several, so a deviation does not stop the run. The deviation itself appears during the avoidance maneuvers. Figure 12 shows this timing: the formation error has its largest deviations in the same interval where the leader angular velocity has its largest swings, and this interval also overlaps with the lowest point of the minimum inter-group distance. The leader turns while the groups are closest, the follower cannot fully match these turns, and after the groups separate the angular velocity returns to near zero and the formation distance decays back toward the target.

The early avoidance also reverses the safety comparison as the scene becomes more crowded. The safe ratio is measured over successful episodes only. At ten groups the expert succeeds in 4% of episodes, and none of those runs stay clear of the 0.4 m collision distance for the whole episode, so its safe ratio is 0%. This matches the average minimum inter-group distance in Figure 10(c), which approaches 0.4 m from four groups on. The expert reaches a goal in a dense scene only by passing other groups with little room left. The GNN turns before the groups are close, so the runs where it succeeds keep a wider gap, and its safe ratio is higher than the expert's from five groups onward.

These results show that the GNN generalizes to scenes denser than the ones it was trained on. The GNN is trained with at most six groups, but it still succeeds with seven to ten. The reason is that each group only reacts to the groups near it, not to the total number of groups in the scene. The avoidance rule it learned on six groups therefore still works when there are more. The GNN has learned a local rule, not a fixed plan for a fixed number of groups. The growth of the mean formation error with the count follows from the same maneuvers. More groups produce more close encounters along the way, each encounter brings another sequence of avoidance turns with the follower lagging, and the formation returns to the target only between encounters. The mean formation error rises as the scene becomes denser.

5.4 Discussion

The two scenarios show the property of the soft-constraint design from two sides. In the single-group scenario, the expert holds the formation as a hard constraint and reaches every goal exactly, while the GNN treats the formation as a soft term and can only approximate it. For long objects, the GNN's errors grow until it can no longer meet the success tolerance, so its success rate drops while the expert stays at 100%. It overshoots or breaks formation, adjusts, and runs out of time while still moving. This happens also because for long objects, training data is sparse and the network has seen fewer similar cases.

In the multi-group scenario, the expert's hard constraints make each group drive straight at its goal before it meets other groups, and a group turns only when it senses another group close by. This works when there are few groups. But when there are more groups, one group meets many others from several directions at once, and there is no space left to turn. In this case the solver either reports infeasibility or holds the groups in place, so the expert fails in most dense scenes. The GNN behaves differently. Its loss terms reward keeping a margin from other groups, so during training the policy is pushed to leave space. And each group exchanges messages with its neighbors within the communication range, so it senses another group while that group is still far away and starts to adjust before the gap gets tight. The expert has neither of these. It drives straight at its goal and turns only when a constraint is about to break, which in a crowded scene is already too late.

The GNN is not perfect either. As shown in the analysis, a formation deviation appears while the groups maneuver around each other, when the follower cannot fully match the leader's avoidance turns. But the deviation does not stop the motion, and the groups keep moving and reach their goals where the expert cannot. The same soft term that costs precision but preserves mobility. So, a failure of the GNN means the robots kept moving but never got precise enough to meet the tolerance. A failure of the expert means the robots stopped and never reached the goal at all.

We may ask whether the expert could reach the same effect simply by loosening its constraints, for example by widening the angle or formation bounds in the CBF conditions so that the safe set is larger and the QP is feasible more often. Loosening the bounds would indeed reduce the deadlocks, since a larger safe set is easier to satisfy when many groups meet. But it may not match the GNN. The loosened bounds are fixed and global. A bound wide enough to keep the expert feasible in a ten-group scene is wider than a single group needs, so the same setting that helps in dense scenes wastes precision in sparse ones. The GNN's deviation is concentrated in the avoidance maneuvers. The formation stays close to the target before the groups meet and decays back toward it after they pass, so the precision cost is paid only where the groups interact. Also, loosening the bounds trades away safety and precision directly. A wider bound means the groups are allowed to come closer or always hold the formation less exactly, not only during the encounters.

The two methods also differ in computation. The expert solves a quadratic program at every control step, and the program grows as more groups and constraints are added. The GNN runs one forward pass per step, and its cost changes little with the number of

groups, while the QP grows as more groups and constraints are added. By construction, the per-step cost of the expert increases with the number of groups.

Our approach has several limitations. First, the safety and formation requirements are soft, so they are not guaranteed. In a crowded or unusual scene, the policy can violate them, and there is no bound on how large the violation can become. The expert does not have this problem. When the QP finds a solution, all constraints hold. Second, the policy depends on the amount and the spread of the training data. The multi-group model is trained on about 800 expert trajectories, extended to roughly 2000 through five rounds of DAgger. More rounds may help, but the training time and the data size are already near the practical limit of our setup. The data is also uneven across group counts. The expert is slow and often fails at seven or more groups, so the dataset contains little successful data from the densest scenes. More high-density expert data would give the network more of the hardest cases to learn from. Third, the policy learns from the expert, so it inherits the expert's limits. The physics-based loss terms let the network learn more than the expert's exact actions, and this is part of why the GNN avoids the expert's deadlock. But the policy is still less precise than the expert, and the slow recovery of the formation after each maneuver shows the same limit.

6 CONCLUSION

This thesis studied collaborative object transportation, where a group of two robots carries an object to a goal, and multiple groups share a workspace and must avoid one another. The standard QP-CBF controller enforces formation and safety as hard constraints. We investigated whether a learning-based decentralized alternative based on GNN could match it in single-group transport and scale better in multi-group scenes.

The first question asked whether the GNN can learn the single-group task of keeping formation and reaching the goal. Within the training range of object lengths, the GNN reaches the goal in nearly all runs, the same as the expert. Its formation is slightly less precise, because it treats the formation as a soft penalty and not a hard constraint. Beyond the training range, for long objects, the accuracy drops gradually. So, the policy still completes the task with lower precision. The GNN can therefore learn the single-group task, with formation precision as the main cost. The second question is about the expert's deadlock in dense scenes. As the scene gets denser, the expert's success rate falls from 100% at two groups to 4% at eight. The GNN stays above 82% through ten groups. It drives most groups to their goals where the expert cannot. But the success measure here only counts whether the groups reach their goals. It does not count formation, and the GNN's formation error grows as the scene gets denser. So, the GNN mostly solves the deadlock, but it is not perfect and does not succeed in every run. The third question is whether the policy still works with more groups than it saw in training. The model is trained with at most six groups, but its success rate stays above 82% from seven to ten groups. The policy therefore holds up beyond its training range.

The rest of this chapter covers three things. Section 6.1 discusses the limitations of the approach. Section 6.2 explains the parameters of the controller and how they were set. Section 6.3 describes possible directions for future work.

6.1 Limitations

Our approach has several limitations. First, as discussed before, the safety and formation requirements are soft, so they are not guaranteed. In a crowded or unusual scene, the policy can violate them, and there is no bound on how large the violation can become. The expert does not have this problem. When the expert finds a solution, all constraints hold.

Second, the policy depends on the amount and the spread of the training data. The multi-group model is trained on about 800 expert trajectories, extended to roughly 2000 through five rounds of DAgger. More rounds may help, but the training time and the data size are already near the practical limit of our setup. The data is also uneven across group counts. The expert is slow and often fails at seven or more groups, so the dataset contains little successful data from the densest scenes. More high-density expert data would give the network more of the hardest cases to learn from.

Third, the policy learns from the expert, so it inherits the expert's limits. The physics-based loss terms let the network learn more than the expert's exact actions, and this is part of why the GNN avoids the expert's deadlock. But the policy is still less precise than the expert, and the slow recovery of the formation after each avoidance maneuver is another sign that the policy cannot reach the expert's precision.

Finally, all results come from simulation. The robots follow ideal unicycle dynamics, and the states are read directly without sensor noise or delay. On real robots the soft policy may behave differently, and how well it transfers is not tested here.

6.2 Parameter Tuning

The parameter values are listed in Tables 5.1 to 5.3. This section explains how the main parameters were chosen and what effect they had.

Some parameters are set by the system or the hardware.

- $\boldsymbol{L_o}$: Set by the object being carried.
- $\boldsymbol{V_{max}}$ and $\boldsymbol{\Omega_{max}}$: The physical speed limits of the robot. The network output is scaled into this range.
- $\boldsymbol{\Delta t}$, $\boldsymbol{r_{safe}}$, $\boldsymbol{r_{min}}$: Come from the robot and the setup.

The remaining parameters were tuned. Their effects are described below.

- $\boldsymbol{r_{comm}}$ (communication radius): This sets how early a group reacts to others. It is set to 5 m. When it was reduced, a group sensed another group only when it was closer, so it turned later and had less room to avoid it. A larger radius lets the group react earlier, but it also adds distant groups to the graph that are not yet relevant.
- $\boldsymbol{\lambda_{form}}$ and $\boldsymbol{\lambda_{arm}}$ (formation): These weights control how tightly the follower holds the formation. When they are set high, the follower stays close to the leader. But during sharp turns the speed needed to keep up goes past V_{max}, the follower

cannot follow, the loss grows very large, and the follower moves erratically. λ_{form} is kept the largest of all the weights, because the formation must be held firmly.

- $\boldsymbol{\lambda_{safety}}$ (safety): This weight controls how strongly the groups avoid each other. It is kept the second largest, since the safety distance must also be held firmly. When it is set too high, the groups turn away fast and hard, the follower falls behind, and the formation suffers.
- $\boldsymbol{\lambda_{goal}}$ and $\boldsymbol{\lambda_{head}}$ (navigation): These weights are set lower than λ_{form} and λ_{safety}. Reaching the goal allows detours, so a group can take a longer path to avoid others or hold its formation. If they were set higher than the formation and safety weights, the groups would head straight for the goal at the cost of the behaviors that must be kept. Heading is already close once the other terms are satisfied, so λ_{head} does not need a large value.
- $\boldsymbol{\lambda_{stop}}$ and $\boldsymbol{k_v}$ (stopping): λ_{stop} controls how a group slows down and settles at the goal. It also gives the linear speed a reference. A larger k_v keeps the group fast while far away and slows it only near the goal. k_v is kept small, at 0.1, so the group moves at a moderate speed and has time to react, avoid other groups, and hold its formation.
- **Learning rate**: Set to 1.5×10^{-4}. A larger rate made training unstable; a smaller one made it slow.
- **Hidden dimension and attention layers**: The network uses 128 hidden units and three attention layers. A smaller hidden dimension has less capacity to represent the interactions between groups, while a larger one increases the number of parameters and the training cost. 128 was chosen as a balance and follows a common size for this kind of network.
- **DAgger rounds**: Set to five. By the last round the gains were small, so the policy had largely converged.

6.3 Future Work

There are several directions for future work follow from the limitations discussed above.

The most direct one is to use another training method. The policy is trained to learn from the QP-CBF expert, so it inherits the expert's limits and struggles to become better than the data it gets from. This matters here because the expert itself is weak in dense scenes, where it frequently deadlocks. A natural next step is to transition from imitation learning

to reinforcement learning. Instead of relying on expert demonstrations, the robots could learn through trial and error to maximize a reward function. To handle the difficulty of dense scenes, the training could follow a curriculum, starting with just two groups and gradually scaling up to more [13]. This would allow the policy to discover successful coordination strategies and faster formation recovery in crowded environments where the current expert fails.

Another major direction follows from the issue of soft constraints. Because the safety and formation requirements in our policy are soft, they are not mathematically guaranteed and can be violated in unusual or crowded scenes. One way to add a guarantee is to learn a safety layer with a graph neural network. In the Control Admissibility Model (CAM) [14], a GNN learns which actions are safe and which are not, so the robot can only pick from the safe set. The robot then plans toward its goal but is restricted to picking a velocity from this safe set. A useful property of this approach is that the model can be trained with only a few robots and then used for much denser scenes, which fits the generalization behavior we observed in our own results.

Additionally, instead of replacing the expert entirely, future work could focus on improving the QP-CBF algorithm itself to resolve its infeasibility issues. The expert often deadlocks because its hard safety and formation constraints conflict in tight spaces. A promising solution is to use a GNN to dynamically tune the internal parameters of the control barrier function online[15]. By sensing how crowded the environment is, the network could relax the less critical constraint margins when groups meet and tighten them again when the scene is sparse. This would help the solver stay feasible in dense scenes and reduce the deadlock, while keeping the hard safety guarantee that the expert provides.

REFERENCES

[1] D. Pedraglio O'Hara and A. Gusrialdi, "Control design for collaborative object transportation using pairs of differential-drive robots via control barrier functions," *Adv. Robot.*, vol. 40, no. 7, pp. 351–365, Apr. 2026, doi: 10.1080/01691864.2026.2626393.

[2] I. Muhammed, A. A. Nada, and H. El-Hussieny, "Real-time decentralized model predictive control for cooperative multi-robot object transport: experimental validation," *Sci. Rep.*, vol. 16, no. 1, p. 9824, Mar. 2026, doi: 10.1038/s41598-026-41881-w.

[3] S. Ghosh, S. Nath, S. Sortee, L. Kumar, and T. Bera, "On the Collaborative Object Transportation Using Leader Follower Approach," in *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Honolulu, Oahu, HI, USA: IEEE, Oct. 2023, pp. 4220–4227. doi: 10.1109/SMC53992.2023.10394439.

[4] A. Khan, V. Kumar, and A. Ribeiro, "Large Scale Distributed Collaborative Unlabeled Motion Planning With Graph Policy Gradients," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5340–5347, Jul. 2021, doi: 10.1109/LRA.2021.3074885.

[5] C. Jiang, X. Huang, and Y. Guo, "End-to-end decentralized formation control using a graph neural network-based learning method," *Front. Robot. AI*, vol. 10, p. 1285412, Nov. 2023, doi: 10.3389/frobt.2023.1285412.

[6] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009, doi: 10.1109/TNN.2008.2005605.

[7] J. H. Tanis, C. Giannella, and A. V. Mariano, "Introduction to Graph Neural Networks: A Starting Point for Machine Learning Engineers," Dec. 27, 2024, *arXiv*: arXiv:2412.19419. doi: 10.48550/arXiv.2412.19419.

[8] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. Wiltschko, "A Gentle Introduction to Graph Neural Networks," *Distill*, vol. 6, no. 8, p. 10.23915/distill.00033, Aug. 2021, doi: 10.23915/distill.00033.

[9] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *International Conference on Learning Representations (ICLR)*, 2018. Available: https://openreview.net/pdf?id=rJXMpikCZ

[10] S. Ross, G. J. Gordon, and J. A. 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*, in Proceedings of Machine Learning Research, vol. 15. PMLR, 2011, pp. 627–635. Available: https://proceedings.mlr.press/v15/ross11a.html

[11] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations (ICLR)*, 2017. Accessed: Jun. 01, 2026. Available: https://openreview.net/forum?id=SJU4ayYgl

[12] G. Li, M. Muller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs Go As Deep As CNNs?," in *2019 IEEE/CVF International Conference on Computer Vision*

(ICCV), Seoul, Korea (South): IEEE, Oct. 2019, pp. 9266–9275. doi: 10.1109/ICCV.2019.00936.

[13] G. Eoh and T.-H. Park, “Cooperative Object Transportation Using Curriculum-Based Deep Reinforcement Learning,” *Sensors*, vol. 21, no. 14, p. 4780, Jul. 2021, doi: 10.3390/s21144780.

[14] C. Yu, H. Yu, and S. Gao, “Learning Control Admissibility Models with Graph Neural Networks for Multi-Agent Navigation,” in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, in Proceedings of Machine Learning Research, vol. 205. PMLR, 2023, pp. 934–945. Accessed: Jun. 17, 2026. Available: https://proceedings.mlr.press/v205/yu23a.html

[15] Z. Gao, G. Yang, and A. Prorok, “Online Control Barrier Functions for Decentralized Multi-Agent Navigation,” in *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, Boston, MA, USA: IEEE, Dec. 2023, pp. 107–113. doi: 10.1109/MRS60187.2023.10416796.