Competitive Multi-Agent Air Hockey Reinforcement Learning Project

Comprehensive Technical Report

Dvir Cohen, Gal Sabo

Executive Summary

This report is a technical analysis of an environment that simulates fast-paced multi-agent air hockey games, as well as several agent policies including reinforcement learning-based and search-based approaches. The project architecture combines a physics simulation engine with the Robot Operating System (ROS) framework for running and evaluating agents, while reinforcement learning agents are trained through repeated simulation.

Proximal Policy Optimization was used as the reinforcement learning method for developing agents capable of scoring while maintaining defensive positioning. Trained agents showed competitive performance in testing, with varying win rates depending on the opponent type.

A separate planning-based approach was developed using Monte-Carlo Tree Search with UCT to support long-short tactical decisions during gameplay. These agents do not require training and instead select actions by evaluating future states within the simulation. In testing, this method achieved competitive results as well, with varying win rates depending on the opponent type and team sizes.

The system supports running different control policies in identical gameplay scenarios, allowing direct comparison between different approaches. This report discusses the strengths and behaviours of several policies, as well as comparative results.

Video Showcase

https://www.youtube.com/watch?v=uOAevuGi2G4

Problem Statement and Objectives

Background and Motivation

Developing agents that can play competitive air hockey introduces several challenges. The environment is fast-paced, physics-driven, and requires continuous control, making both offense and defense tightly coupled in time. Agents must position themselves effectively, react to puck motion, and prevent scoring attempts while maintaining the ability to strike toward the opponent’s goal.

These requirements become more complex in a multi-agent setting, where teammates must coordinate without obstructing each other, and opponents create pressure by contesting space and influencing puck trajectories. The need to balance quick reaction with anticipating future puck behavior motivates exploring different decision-making approaches. This project therefore examines reinforcement learning for learned strategic behaviors, as well as search-based planning for real-time tactical decisions.

Project Objectives

The project established several objectives to guide system development and evaluation. First, it required a simulation environment capable of running competitive multi-agent air hockey games and supporting efficient testing. Second, reinforcement learning agents needed to be trained to handle both offense and defense while maintaining coordination with teammates. Third, integration with the Robot Operating System (ROS) was needed to allow real-time agent execution and potential future deployment. Additionally, the system needed to support alternative decision-making approaches, enabling comparison between trained policies and search-based planning methods. Finally, the project aimed to evaluate performance across multiple scenarios to understand the strengths and limitations of each approach.

Environment and System Architecture

Physics Simulation

The system includes a physics-based simulation used to run competitive multi-agent air hockey games. Agent paddles and the puck are modeled as rigid circles with configurable radii and mass properties. Circular collision detection is used to determine contact points, and momentum transfer is applied based on relative velocities to produce realistic puck responses. Additional parameters such as puck speed limits, friction, and bounce damping ensure that motion remains stable and visually consistent over time.

The simulation processes player actions at 60 Hz using a fixed time step, which maintains reproducibility and avoids variations in gameplay dynamics across runs. Score tracking, goal detection, and episode termination conditions are managed directly within the simulation, enabling consistent evaluation of different agent strategies.

ROS Coordination Framework

The Robot Operating System (ROS) middleware is used to manage communication between the simulation and agent policy nodes. Each agent runs as an independent node, receiving state observations and publishing control commands. A central Game Manager node initializes episodes, distributes state updates, and handles scoring and resets. This structure supports running multiple match configurations and enables the same environment to be used for policies based on different approaches.

Agents do not communicate among themselves or with teammates. To ensure comparable behaviour between the two teams, the game manager node mirrors the world state for the agents on team B so that the same objective logic applies symmetrically on both sides of the field.

Game Management and Scripted Agents

In addition to learning-based and planning-based policies, the system supports scripted agents that execute predefined behavior without using model inference or communication. These agents serve as simple opponents or teammates for controlled testing of more advanced policies. The Game Manager ensures consistent match flow, initialization, and termination conditions across all policy types.

Reinforcement Learning Approach

Policy Network Architecture

The neural network architecture implements a shared feature extraction backbone followed by separate actor and critic heads for policy and value function approximation. The network processes normalized observations including agent positions, puck state with position and velocity, and relative positions of teammates and opponents. This architecture enables efficient learning while maintaining the computational efficiency necessary for real-time inference during gameplay.

Proximal Policy Optimization Implementation

The training methodology employs Proximal Policy Optimization, a policy gradient algorithm known for its stability and sample efficiency in continuous control tasks. The implementation maintains separate policy and value networks sharing a common feature extraction backbone, enabling efficient learning while preventing catastrophic policy updates using clipped surrogate objectives.

The algorithm collects experience through rollouts of fixed length, accumulating state observations, actions, rewards, and values estimates. After each rollout phase, the system computes advantage estimates using Generalized Advantage Estimation with configurable discount and lambda parameters. The policy and value networks then undergo multiple epochs of minibatch updates, with gradient clipping to prevent excessively large parameter updates that could destabilize learning.

Key hyperparameters were selected through systematic experimentation to balance exploration and exploitation while maintaining training stability.

Reward Shaping and Incentive Design

The reward structure implements a multi-component design to encourage desired behaviours while discouraging suboptimal strategies. Goal scoring provides the primary reward signal with a magnitude of one thousand, creating a strong incentive for offensive play. Contact with the puck yields a reward of one hundred, encouraging agents to actively engage with gameplay rather than remaining passive.

Dense shaping rewards guide agents toward productive behaviours during the learning process. Proximity rewards scale with inverse distance to the puck, encouraging agents to position themselves advantageously. Shooting rewards provide additional incentive when agents strike the puck toward the opponent's goal, with magnitude proportional to the alignment between puck velocity and goal direction. Defensive positioning rewards encourage agents to maintain coverage of their own goal when the puck is in dangerous positions.

Penalty terms discourage undesirable behaviours such as excessive collisions with teammates and unnecessary movement when far from the action. These penalties help guide agents toward efficient strategies while preventing the emergence of pathological behaviours that could interfere with team coordination.

Curriculum Learning Approach

The project implements a progressive curriculum learning methodology to facilitate skill acquisition across multiple difficulty levels. This approach structures training into discrete phases, each focusing on specific competencies while gradually increasing task complexity. The curriculum design enables agents to master fundamental behaviours before progressing to more sophisticated strategies.

The initial phase focuses exclusively on puck interaction, training agents to approach and contact the puck against static opponents. This phase employs high exploration rates and shorter episode lengths to accelerate the learning of basic motor skills. The reward structure during this phase heavily weights contact events to provide clear learning signals.

Subsequent phases progressively introduce shooting mechanics, defensive positioning, and eventually competitive play against increasingly sophisticated opponents. Each phase transition includes adjustments to hyperparameters such as learning rate and entropy coefficient to account for the changing nature of the learning task. The curriculum approach significantly improves sample efficiency compared to direct training on the full task complexity.

Planning-Based Approach (Long-Short MCTS)

Short-Term Objectives

The planning approach uses a set of short-term scripted objectives, each describing a specific intention such as defending the goal line, intercepting the puck, or executing a fast or pass-oriented shot. These objectives produce simple directional commands without constantly forcing motion, allowing agents to hold position when needed.
Several factors influence how a given objective behaves, including puck proximity, agent position whether a teammate is nearby and some preset factors. Some objectives also include built-in avoidance behaviours to reduce interference between teammates.

Long-Term Decision Making with MCTS-UCT

To select which objectives should be executed, the system employs Monte-Carlo Tree Search with Upper Confidence Bounds for Trees (UCT). MCTS evaluates actions by simulating their future outcomes, while UCT balances the choice between trying promising actions and exploring alternatives.

In each search tree node, the system considers objective assignments for both teams rather than alternating turns. Node selection uses a min-max formulation of the UCT score, accounting for the opposing team's response. The tree expands by forwarding the simulation several steps under the selected objective assignments. Opponent behavior is also modeled using the same set of objectives, regardless of whether they are being executed by a planning-based policy or another type of policy.

Design Considerations

Each tree node represents multiple physics steps executed under a chosen combination of objectives, allowing longer-term outcomes to be estimated in fewer total expansions. Evaluation values are updated throughout the search while gradually reducing the influence of older estimates and prioritizing recent rollouts.

To keep branching manageable, only a limited set of pre-selected objective combinations is used rather than enumerating every possible sequence for every agent. Objectives are assigned to players based on a rule that considers their x-position and an offensive factor specific to each agent. Passing objectives are available whenever at least two teammates are present, and they enforced on lower-priority players (e.g., fifth and onward) to avoid interfering other agents. Evaluation values are not obtained by simulating to the end of an episode. Instead, a heuristic score is applied at each expansion to estimate the resulting state without additional rollout steps.

Challenges and Adjustments

Initially, short-term objectives generated a queue of commands intended to smooth behaviour over multiple frames, but this created delayed paddle responses and reduced reactivity. MCTS evaluation also proved expensive, so the number of simulation steps per expansion was reduced to limit search depth and keep planning time within real-time constraints. Those changes maintained responsiveness while preserving the benefits of look-ahead planning.

Work Distribution and Implementation

Core Simulation Development (Dvir and Gal)

The simulation engine component required implementation of the physics system, game logic, and visualization subsystems. Development involved creating the base engine class that manages game state, processes agent commands, and computes physics updates. The collision detection system was designed to handle multiple simultaneous contacts while maintaining computational efficiency. The visualization system provides real-time rendering of game state using pygame module, enabling visual debugging and demonstration of trained agent behaviours.

ROS Integration and Agent Framework (Dvir)

Integration with the Robot Operating System required developing agent node implementations, game manager nodes, and the message passing infrastructure. The agent nodes encapsulate policy execution, receiving world state observations and publishing action commands. The game manager node coordinates episode initialization, state distribution, and termination detection. Launch files and configuration systems enable flexible deployment across different scenarios and team compositions.

Neural Network and Training Pipeline (Gal)

The machine learning components required implementation of a neural policy architecture, the training loop for Proximal Policy Optimization, and the experience collection system. Development work included designing the network with appropriate input normalization, output activation functions, and layer dimensions. The training pipeline handles rollout collection, advantage computation, policy updates, and checkpoint management. Extensive logging and diagnostic systems enable monitoring of training progress and identification of potential issues.

Curriculum Learning System (Gal)

The curriculum learning extension required creating phase definitions, transition logic, and hyperparameter scheduling systems. Implementation involved developing configuration files that specify training phases with their associated timestep allocations, opponent types, reward weights, and termination conditions. The training orchestrator manages transitions between phases, applying appropriate hyperparameter updates and reward weightings as agents progress through the curriculum.

Long-Short Planning Module (Dvir)

The planning-based control system combines short-term scripted objectives with long-term Monte-Carlo Tree Search using UCT. The short-term objectives define actionable intentions such as defending or shooting and are assigned to agents based on position and role. MCTS-UCT evaluates combinations of these objectives for both teams in real time by expanding a search tree and applying heuristic scoring instead of full rollouts. Implementation included the decision logic, objective selection rules, and tree data structures, with attention to limiting branching and keeping computation time within real-time constraints.

Testing and Evaluation Infrastructure (Dvir)

Comprehensive testing infrastructure enables evaluation of trained policies against various opponent strategies. The testing suite includes evaluation scripts that load trained checkpoints, run episodes against specified opponents, and compute performance statistics. Visualization tools enable qualitative assessment of agent behaviours, while statistical analysis provides quantitative performance metrics.

Conversion and Deployment Tools (Gal)

Tools for converting trained models from training format to deployment format enable seamless transition from development to production deployment. The conversion process extracts learned policy networks from training checkpoints and packages them as standalone policy objects compatible with the ROS agent framework. This separation ensures that deployment environments do not require training dependencies while maintaining full compatibility with trained behaviours.

Results and Performance Analysis

Scenarios and Policies

We evaluate six control policies across four arena configurations (default, small, wide, large) in both 2v2 and 4v4 match formats. Scenarios are defined in the simulation package under
src/air_hockey_ros/game_scenarios.

Scenarios:
Each scenario modifies the environment dynamics by altering table dimensions, friction, and agent/puck speed constraints. This enables evaluation of policy robustness under different spatial and kinematic conditions.

- Default scenario — standard board size; balanced offense–defense gameplay

- Small scenario — reduced field dimensions; faster interactions and less reaction time

- Wide scenario — increased width and added friction (friction_per_tick = 0.995, however, the value represents friction coefficient). encourages lateral play and passing

- Large scenario — increased width & height with friction; longer travel distances and greater defensive coverage difficulty

A shallow rectangular goal zone (width = goal_gap, depth = goal_offset) exists in each scenario. Although the simulation rules do not prohibit entering this area or crossing the half-line, policies are scripted to avoid spending excessive time inside the defensive goal zone or crossing the half-line unnecessarily, to prevent trivial goal-blocking strategies.

The default scenario is used for most evaluation matches. The other configurations assess generalization to different environmental demands.

Policies:
We evaluated two adaptive methods and four scripted baselines.
Adaptive:

- **two_capped_neural (PPO-based neural agent)**
  Uses a neural network trained with proximal policy optimization.
  The observation space supports up to two teammates per side, hence the "two_capped" name. This limitation prevented meaningful evaluation in 4v4, as the observation encoding does not account for all teammates.
  Typical behavior: aggressive puck pursuit from behind, often coordinated unintentionally without collisions.
- **long_short (MCTS–UCT objective planning)**
  Uses short-term objectives for immediate commands, selected via a Monte-Carlo Tree Search with UCT.
  A defensive–offensive role separation often emerges without manual scripting, with occasional dynamic switching depending on state and search outcome.

Scripted:

- **short** - the same objectives as long_short, but with **static role assignments** and no planning or switching
- **free_simple** - pursues puck from behind without teammate awareness
- **regional_simple** - field divided horizontally so each agent remains in its assigned region, reducing interference
- **simple** - similar to regional_simple, but with one defender patrolling a vertical line instead of chasing the puck

Both long_short and short policies utilize the following short-term objectives:

- Defend Line – moves along a vertical line near the team's goal, adjusting laterally to intercept the puck when its trajectory approaches the goal x-coordinate.
- Intercept – actively pursues the puck when it moves toward the defensive half, aiming to reverse its x-axis velocity rather than score.
- Pass Shot – avoids interfering with teammates; if the puck is reachable, attempts an upward pass to another agent, otherwise performs a direct shot.
- Fast Shot – aggressively strikes the puck toward the opponent's goal immediately upon contact, doesn't try to defend goal

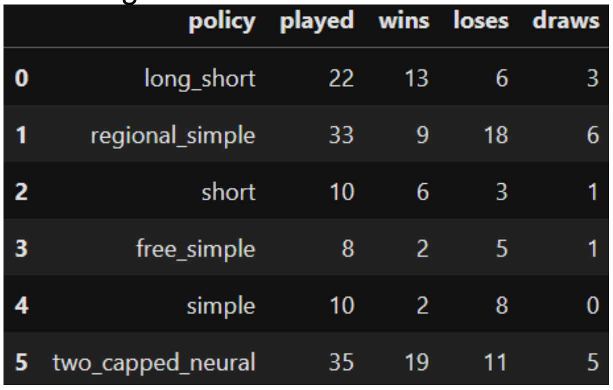
Performance Evaluation

We have simulated 71 games, all under:
game_logs\tournament_games
We evaluate the final team policies — the neural network–based policy and the long-short planning policy — in direct competition against each other and against a variety of scripted baseline agents. Evaluations include multiple scenarios and both 2v2 and 4-vs-4 configurations where applicable, each consisting of two minutes of gameplay at a 60 Hz simulation frequency. Performance is assessed using gameplay outcomes (wins, losses, draws), score differentials, scenario-dependent results, pairwise score matrices, and Elo rankings.

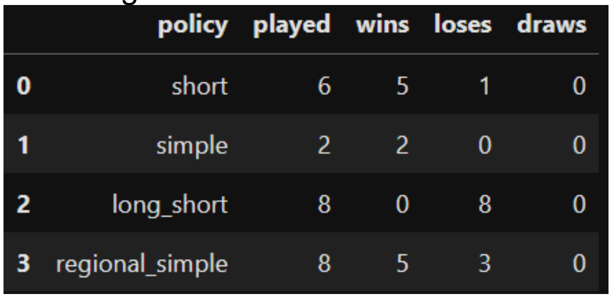
Overall Match Statistics

<u>Those are the play and win / lose / draw counts:</u>

For 2v2 games:



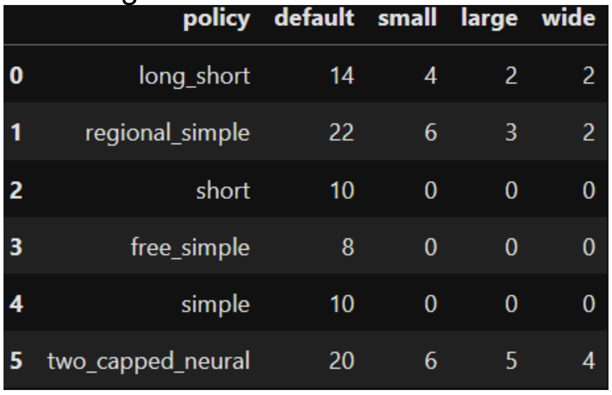
	policy	played	wins	loses	draws
0	long_short	22	13	6	3
1	regional_simple	33	9	18	6
2	short	10	6	3	1
3	free_simple	8	2	5	1
4	simple	10	2	8	0
5	two_capped_neural	35	19	11	5

For 4v4 games:


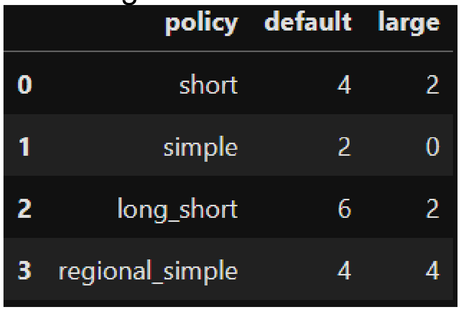
	policy	played	wins	loses	draws
0	short	6	5	1	0
1	simple	2	2	0	0
2	long_short	8	0	8	0
3	regional_simple	8	5	3	0

Policy two_capped_nerual won 54.2% of all games and lost 31.4% of all 2v2 games.
Policy long_short won 59% of all games and lost 27.2% of all games among all 2v2 games. However, it lost all 4v4 games it participated in.
It seems that among the scripted opponents, short is the most capable and free_simple is the weakest.

<u>Those are the play across scenario counts:</u>

For 2v2 games:


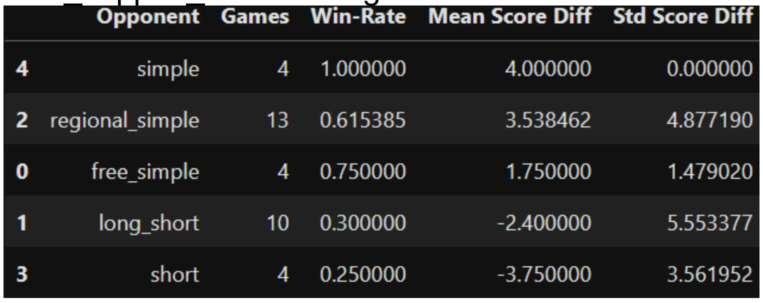
	policy	default	small	large	wide
0	long_short	14	4	2	2
1	regional_simple	22	6	3	2
2	short	10	0	0	0
3	free_simple	8	0	0	0
4	simple	10	0	0	0
5	two_capped_neural	20	6	5	4

For 4v4 games:


	policy	default	large
0	short	4	2
1	simple	2	0
2	long_short	6	2
3	regional_simple	4	4

Performance vs opponents:

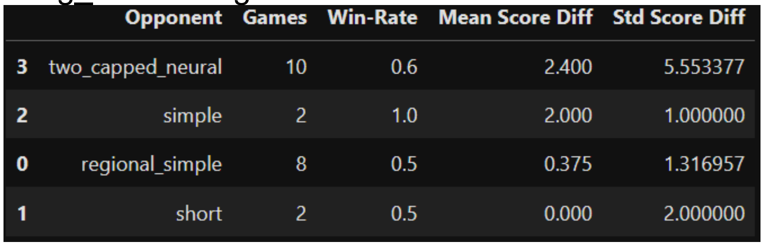
Two_capped_neural 2v2 games:



	Opponent	Games	Win-Rate	Mean Score Diff	Std Score Diff
4	simple	4	1.000000	4.000000	0.000000
2	regional_simple	13	0.615385	3.538462	4.877190
0	free_simple	4	0.750000	1.750000	1.479020
1	long_short	10	0.300000	-2.400000	5.553377
3	short	4	0.250000	-3.750000	3.561952

The neural policy in 2v2 demonstrated strong results against simple and free_simple with high win rate and against regional_simple with high mean score. When facing short and long_short policies, performance declined, indicating sensitivity to opposing more sophisticated opponents.

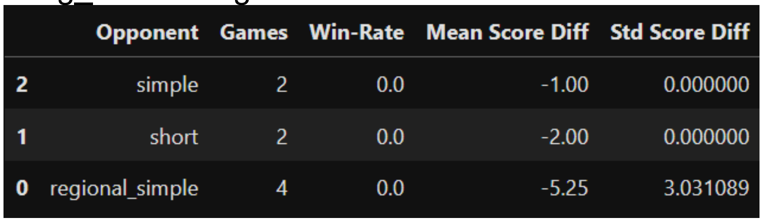
Long_short 2v2 games:



	Opponent	Games	Win-Rate	Mean Score Diff	Std Score Diff
3	two_capped_neural	10	0.6	2.400	5.553377
2	simple	2	1.0	2.000	1.000000
0	regional_simple	8	0.5	0.375	1.316957
1	short	2	0.5	0.000	2.000000

Long _short demonstrated strong results against all opponents in 2v2 games, overperforming the neural policy. It seems that the usage of MCTS UCT didn't help the long_short policy to overperform short policy, as long_short and short performed similarly against each other in the limited direct matches recorded (2 games): each one scored 2-0 against the other by the 2-default scenario 2v2 games they played.

Long_short 4v4 games:

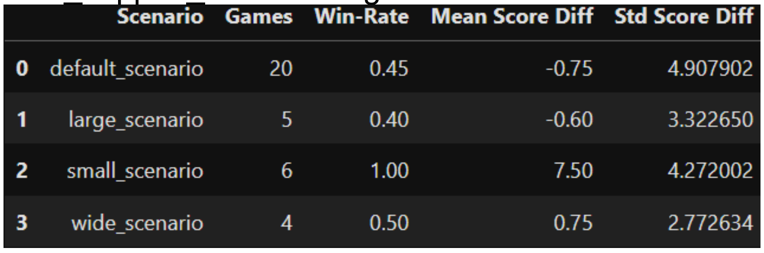


	Opponent	Games	Win-Rate	Mean Score Diff	Std Score Diff
2	simple	2	0.0	-1.00	0.000000
1	short	2	0.0	-2.00	0.000000
0	regional_simple	4	0.0	-5.25	3.031089

The long-short planner struggled in all the 4v4 games, suggesting its model did not scale effectively. Interestingly, it seems regional_simple policy mostly exceled in those settings. This is expected, as regional_simple enforces positional lane separation that remains robust even with more agents.

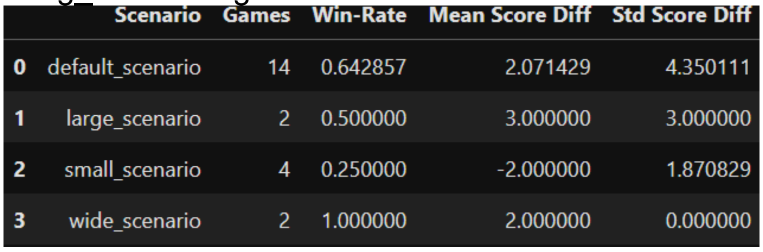
Performance Across Scenarios

Two_capped_neural 2v2 games:



	Scenario	Games	Win-Rate	Mean Score Diff	Std Score Diff
0	default_scenario	20	0.45	-0.75	4.907902
1	large_scenario	5	0.40	-0.60	3.322650
2	small_scenario	6	1.00	7.50	4.272002
3	wide_scenario	4	0.50	0.75	2.772634

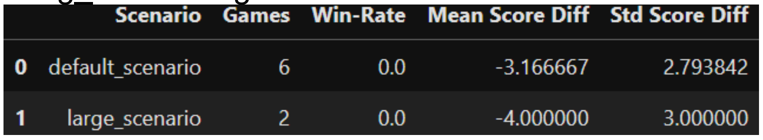
Long_short 2v2 games:



	Scenario	Games	Win-Rate	Mean Score Diff	Std Score Diff
0	default_scenario	14	0.642857	2.071429	4.350111
1	large_scenario	2	0.500000	3.000000	3.000000
2	small_scenario	4	0.250000	-2.000000	1.870829
3	wide_scenario	2	1.000000	2.000000	0.000000

In 2v2, both neural and long-short policies maintained competitive behaviour across arena sizes, with two_capped_neural succeeding in small scenario while long_short succeeding in wide scenario.

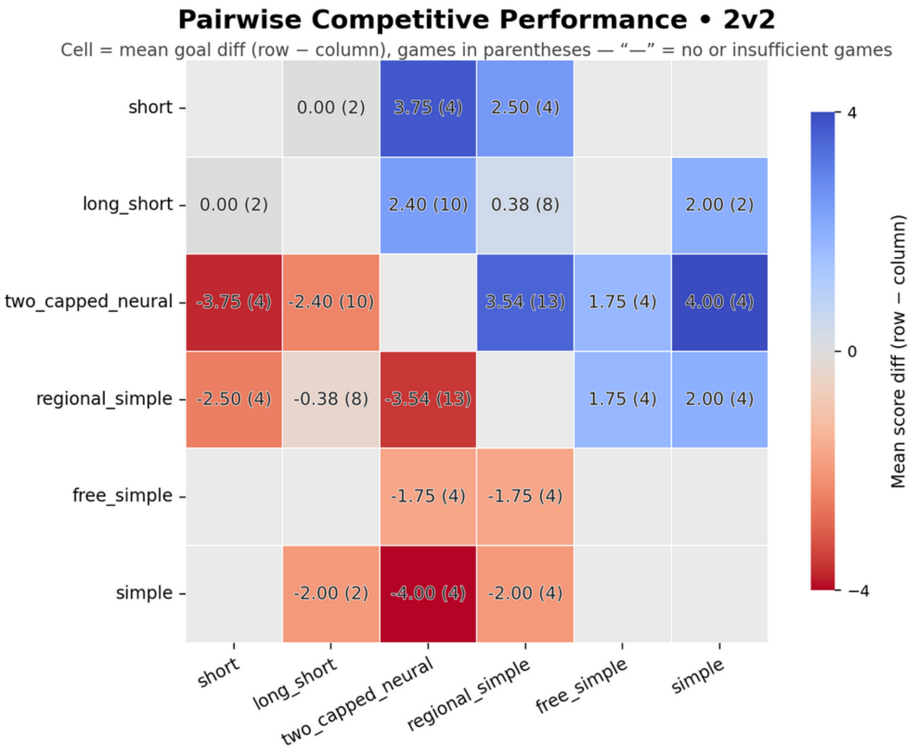
Long_short 4v4 games:

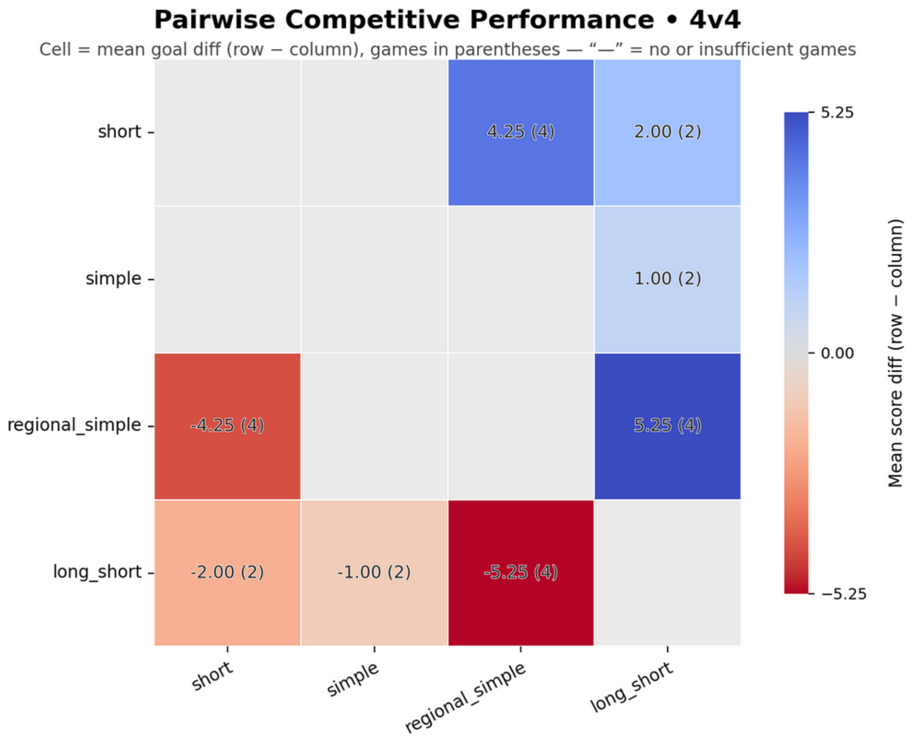


	Scenario	Games	Win-Rate	Mean Score Diff	Std Score Diff
0	default_scenario	6	0.0	-3.166667	2.793842
1	large_scenario	2	0.0	-4.000000	3.000000

In 4v4, long-short performance deteriorates regardless of board size.

Pairwise Score Matrix

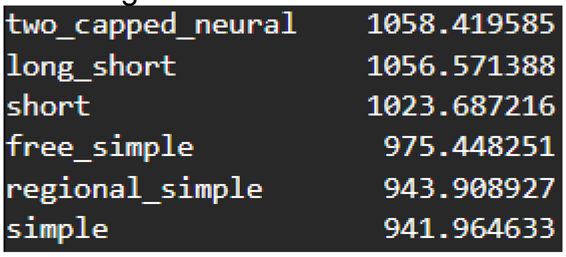




In 2v2, both the neural team policy two_capped_neural and the long-short policy outperform the scripted baselines, with the neural agents excelling offensively and long-short showing slightly better strategic control in direct matchups. However, in 4v4, long-short struggles to scale its coordination model, performing worse than scripted baselines and indicating that its planning advantage does not fully extend to larger team dynamics.

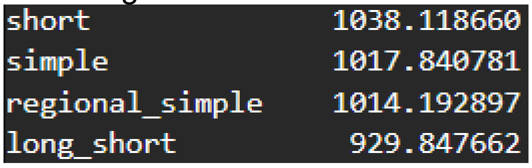
ELO ranking

Elo ranking provides a single competitive score for each policy by updating ratings based on expected versus actual match outcomes, allowing fair comparison across all opponent matchups.
For 2v2 games:



```
two_capped_neural    1058.419585
long_short           1056.571388
short                1023.687216
free_simple           975.448251
regional_simple       943.908927
simple                941.964633
```

For 4v4 games:



```
short                1038.118660
simple               1017.840781
regional_simple      1014.192897
long_short            929.847662
```

As observed in the score matrices, the neural agent and the long-short planner remain the strongest strategies in 2v2 based on Elo ratings, while the long-short planner falls below scripted baselines in 4v4. Additionally, the Elo gaps highlight how sharply the competitive ordering shifts between formats: 2v2 presents a clear top tier of adaptive agents, whereas 4v4 flattens the field and even reverses strengths, suggesting that long-range planning may become a liability when rapid local coordination dominates gameplay dynamics.

Behavioural Analysis

Neural Policy (two_capped_neural)

The neural agents exhibit highly aggressive puck-chasing behaviour, typically approaching from behind the puck to maintain forward momentum toward the opponent goal. While each agent maintains its own independently trained weights, they share the same network architecture, which leads to similar decision tendencies and coordinated pursuit patterns.

During defensive transitions, the agents often withdraw with the puck rather than intercepting or redirecting it early, occasionally allowing deeper penetration into the defensive zone than optimal. Their spatial awareness is limited, and they tend to drift toward boundary regions. They would also cross the half-line into the opponent's side if their own policy logic did not explicitly suppress such actions. Despite these limitations, their continuous pressure generally reduces the frequency of uncontested scoring attempts and produces strong offensive engagements in 2v2 play.

Long-Short Planner (long_short)

The long_short agents implement a short-term objective framework with selection driven by MCTS–UCT. In practice, this produces behaviours that appear role-differentiated:
in 2v2 matches, one agent typically functions as a defender, maintaining a vertical defensive line and intercepting incoming trajectories — even when the puck approaches near the walls rather than directly through the goal gap. This defender generally prefers back-of-puck contact, which contributes to more stable clearances.

The second agent adopts a more offensive, fast-shot–oriented role, attempting quick returns toward the opponent goal. Its reaction is not always immediate, but it demonstrates awareness of puck approach direction — occasionally stepping aside to allow the puck to pass behind before striking, which is tactically advantageous.

Compared to the 'short' scripted baseline, which maintains strictly static roles, long_short exhibits occasional objective switching: the defending agent may briefly advance before returning to its defensive line, and the offensive agent may fall back to support defence when necessary. These adaptations, though not frequent, indicate that UCT-based planning introduces greater situational flexibility than purely scripted behaviour.

In 4v4, role structure becomes more emergent and dynamic. Agents often organize into a staggered formation — three moving toward contesting the puck while one remains as a deep goalkeeper — achieving spatial coverage without explicit regional constraints. Some intra-team interference still occurs due to the lack of strict positional boundaries, but the level of coordination is notable given that lane assignment is not manually imposed. Objective switching is slightly more frequent in this format, though the increased agent density leads to occasional congestion and diluted role clarity.

Conclusion and Future Directions

Summary of Achievements

This project demonstrates the potential of both reinforcement learning and search-based planning approaches in a competitive multi-agent robotic environment. The neural PPO-based policy achieved competent offensive behaviour and reliable puck pressure. In parallel, the long-short planner showed that objective-focused reasoning with MCTS-UCT can produce role-coordinated behaviour and strong defensive structure in 2v2 play.

The system was developed on top of the **Robot Operating System (ROS)** framework, ensuring modularity and enabling future transfer to physical robotic platforms. Simulation, policy execution, and logging were kept separate through a clean architecture, allowing both rapid experimentation and integration with real-time control pipelines.

The comprehensive evaluation across four arena configurations and multiple scripted baselines provides meaningful insights into team coordination, scalability, and decision quality under different gameplay demands.

Key conclusions

Analysis of performance revealed important practical findings:

- Planning adds value: UCT-based search improves organization and defence — as long as decision latency remains small.
- Short-term objectives matter: Having explicit roles produces better coordination than fully reactive methods.
- Scaling remains difficult: The long-short policy struggled when moving from 2v2 to 4v4, showing that one search step per frame was insufficient for complex multi-agent interactions.
  Likewise, adjusting the PPO training process for more agents proved time-consuming, and achieving stable learning at larger team sizes remains an ongoing challenge.
- Neural vs Planning trade-offs: Neural agents excel offensively, while planners provide more structured positioning.

Future Research Opportunities

Building on these insights, several directions can enhance the system:

- Faster MCTS value estimation: Use neural heuristics to evaluate future states, reducing rollout cost and enabling deeper search each frame.
- Improved objective logic: Incorporate teammates and opponent predictions to reduce interference and late defensive reactions.

- Hybrid planning-and-learning agents: Train neural sub-policies specialized for objectives (intercept, fast-shot, etc.), executed within the UCT planner.
- Better scalability strategies: Coordination mechanisms and lane management to prevent 4v4 congestion and role collapse.
- Transfer to hardware: Domain randomization and calibration techniques to close the simulation-to-reality gap.

Practical Applications

The methods demonstrated in this project, such as multi-agent coordination, real-time search, and training-based control have relevance beyond air hockey:

- cooperative robots in logistics,
- autonomous vehicle formation control,
- defensive intercept systems,
- distributed swarm robotics.

The ROS-based integration ensures a realistic pathway to deployment in these domains.

Conclusion

This project shows that objective-driven planning and modern reinforcement learning can both produce competitive multi-agent performance in a dynamic high frequency robotic environment. While scalability challenges remain, the work establishes a strong foundation for future hybrid systems in which planning provides structure and interpretability and learning supplies adaptability and speed.