Multi-Agent RL in a competitive Tron light-Cycle Environment

Project

Presented by
Erick Buitrago
Juan Palomino

Problem Statement

- Tron is a competitive game where players control light cycles that leave trails. The goal is to avoid collisions and force the opponent to crash.
- Training intelligent agents for this environment requires fast decision-making and strategic behavior in a multi-agent setting.

General Objective:

Design and implement reinforcement learning agents to compete in the Tron light cycle game, aiming to test intelligent behavior in dynamic adversarial environments using PPO in RLlib.

Specific Objectives:

- Analyze the Tron game system to understand its dynamics and structure, enabling an effective problem formulation.
- Build a custom training environment with proper state, action, and reward definitions to simulate the game.
- Train agents using PPO in RLlib, allowing adaptation to opponents and learning strategic behaviors.
- Evaluate agent performance using metrics such as win rate, survival time, and movement efficiency.

System design:

- Uses Cases.
- Functional Requirments.
- Non-Functional. requirments.
- Environment 2D grid.
- RL System Design

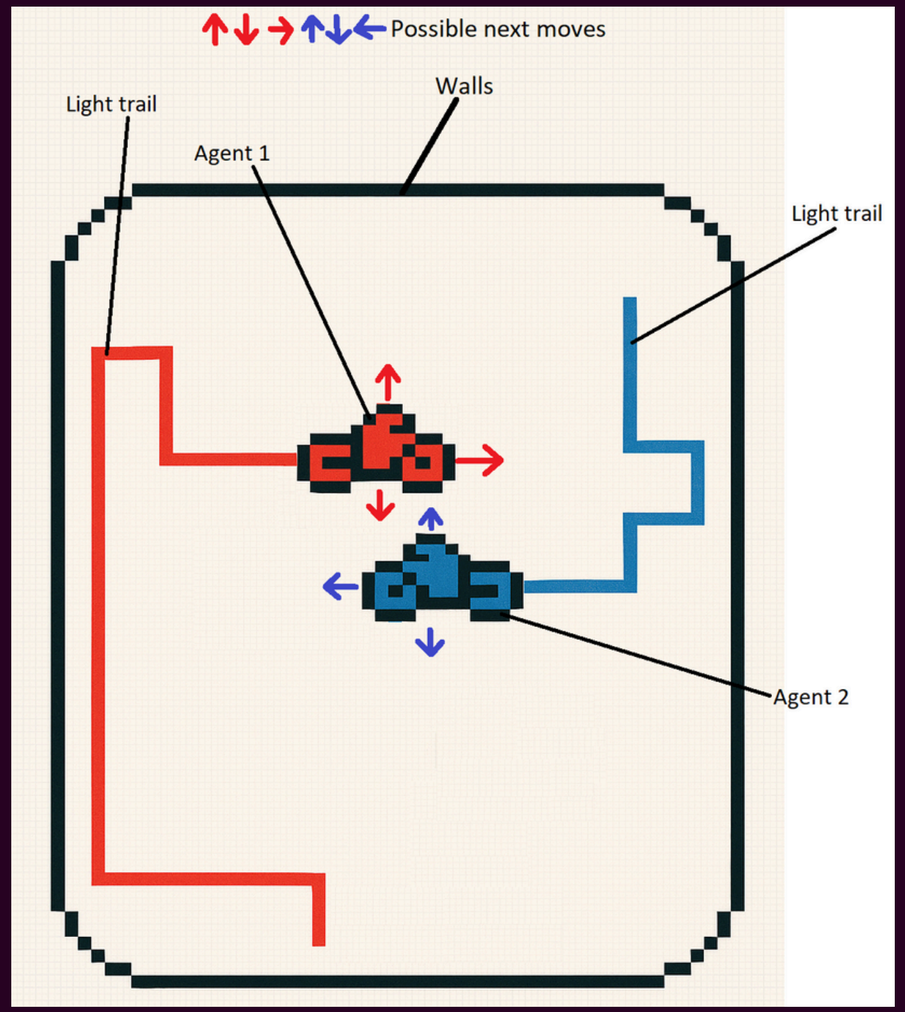
Use Cases

01 Move in the map

02 Toogle light trail

03 Avoid obstacles

04 Cause opponent to crash



Functional Requirements

01 Generate rewards

02 Support multiple reinforcement learning algorithms

03 Create an environment for the training

04 Visualize agent movement and trail generation in real-time

05 Visualize and analize metrics and performance

Non-functional requirements

01 The observation tensor must be updated on every step

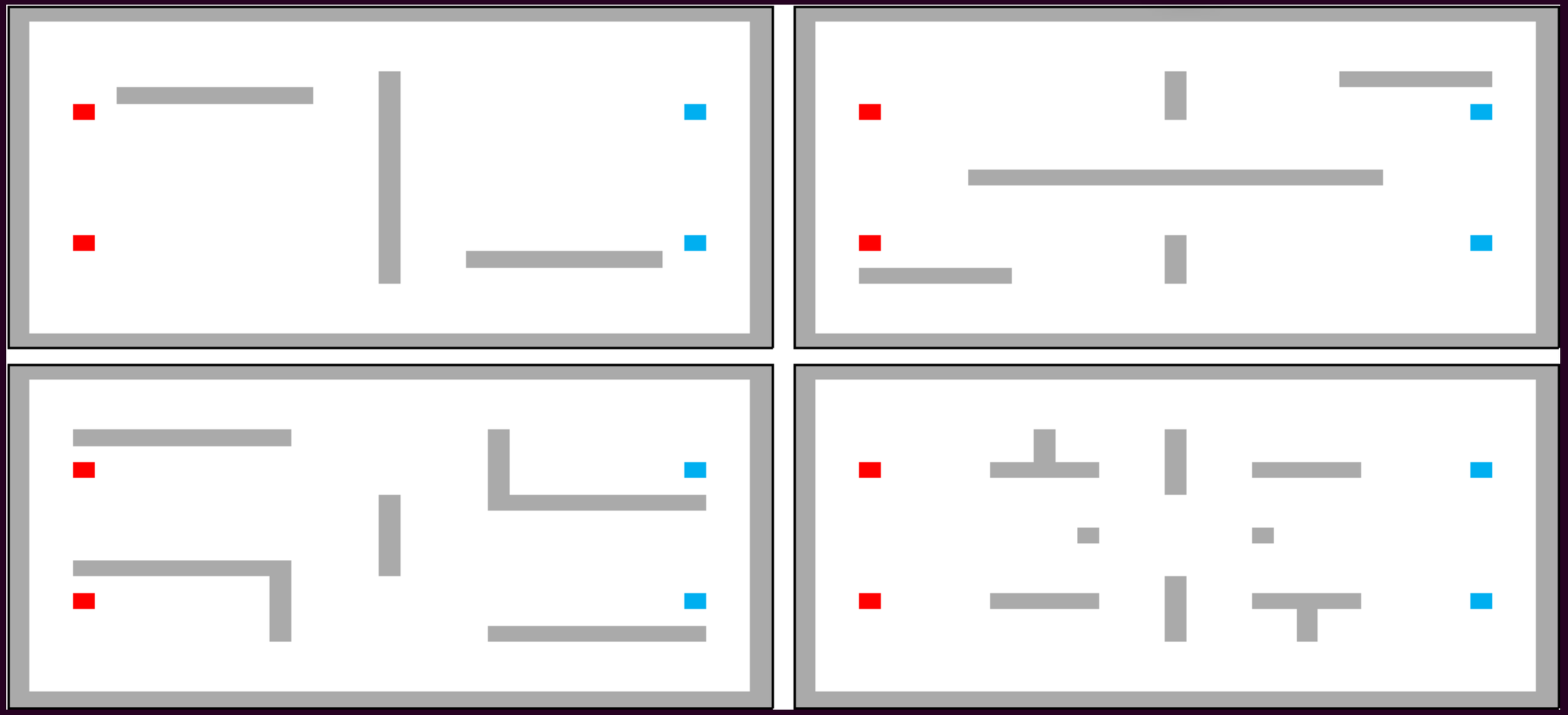
02 The training process should be reproducible across multiple seeds.

03 Allow configuration of reward parameters to test different agent behaviors.

04 The environment should be lightweight and executable on standard GPU hardware.

05 All logs and metrics should be stored for post-training analysis and comparison.

Environment 2D grid:

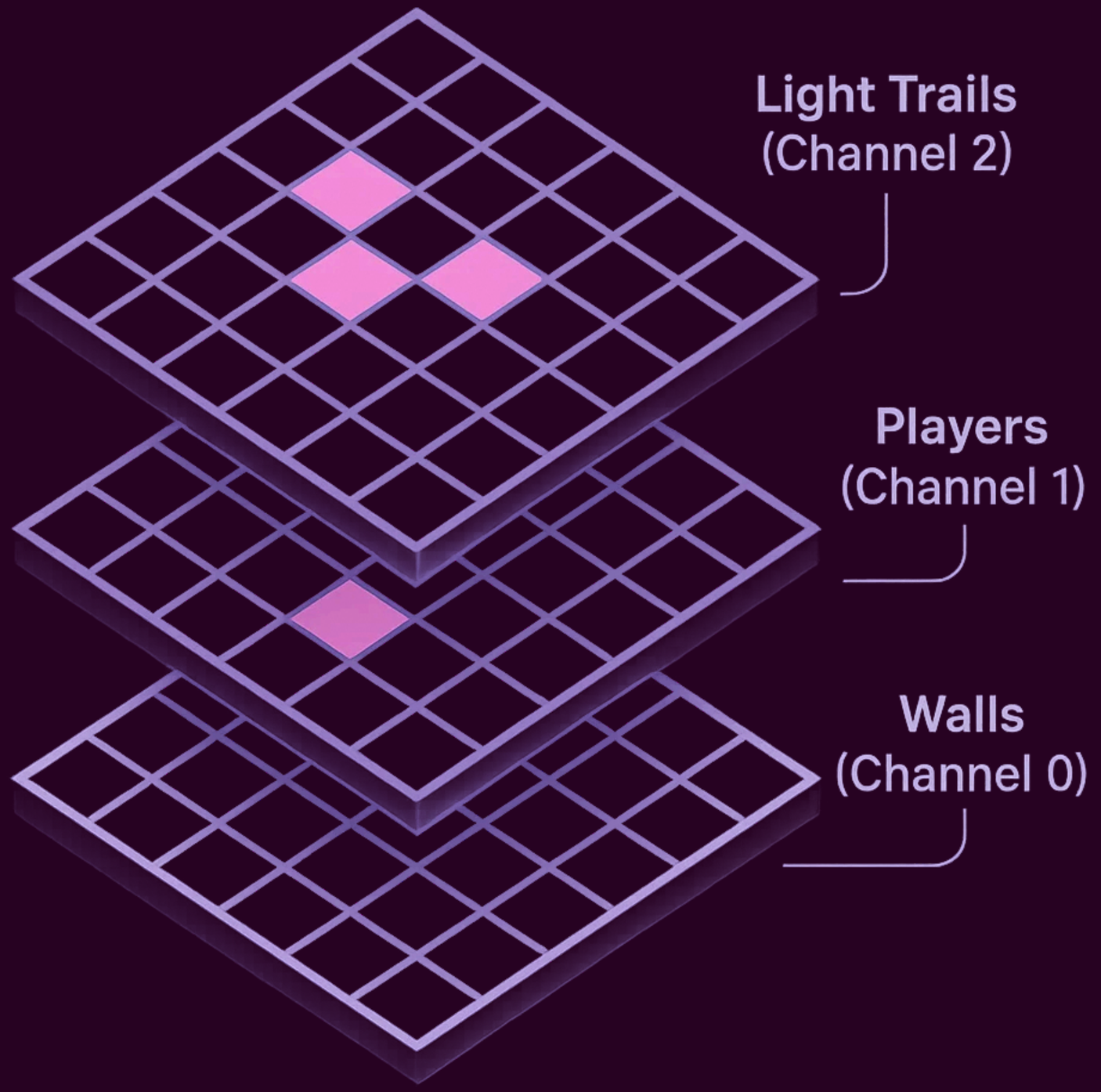


Reinforcement Learning System Design

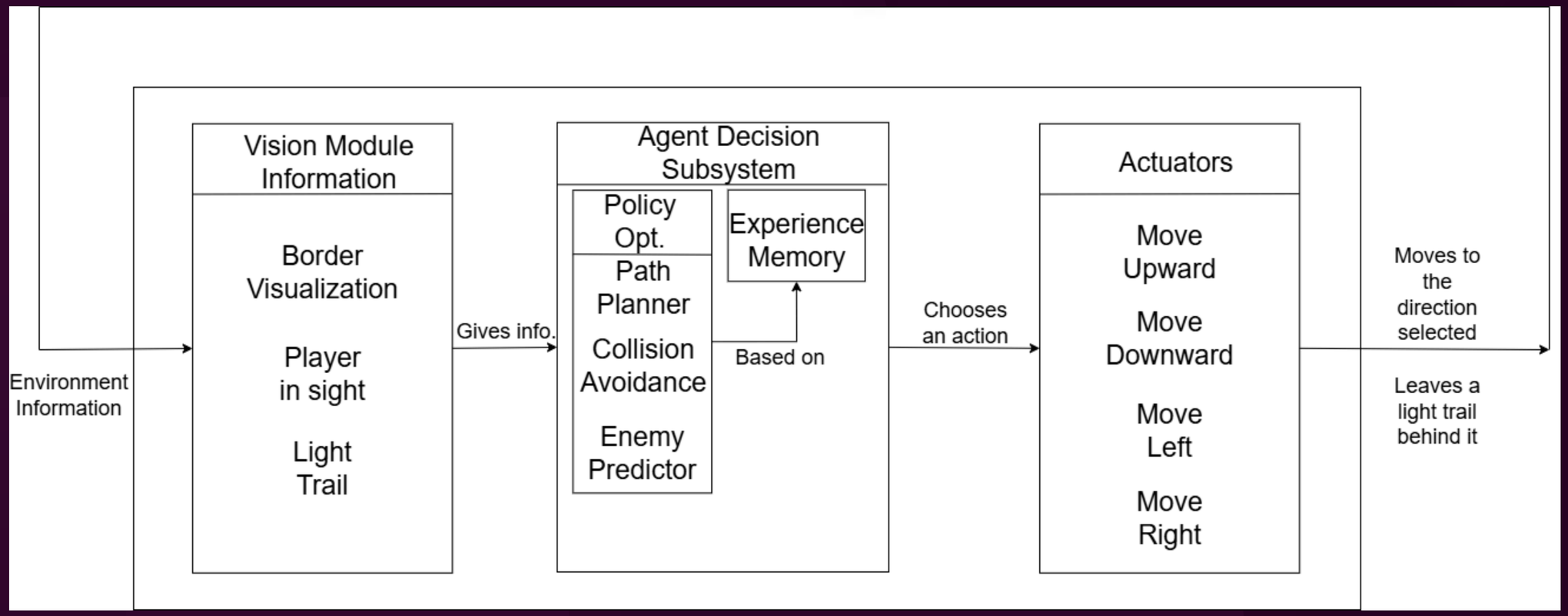
Sensors	Vision cone gathers positions, trails, walls.
Actuators	Discrete movement and light trail toggle.
Perception	Agent observes obstacles, opponents, and map limits.
Actions	Up, down, left, right.
Reward system	Encourages survival and aggressive behavior.

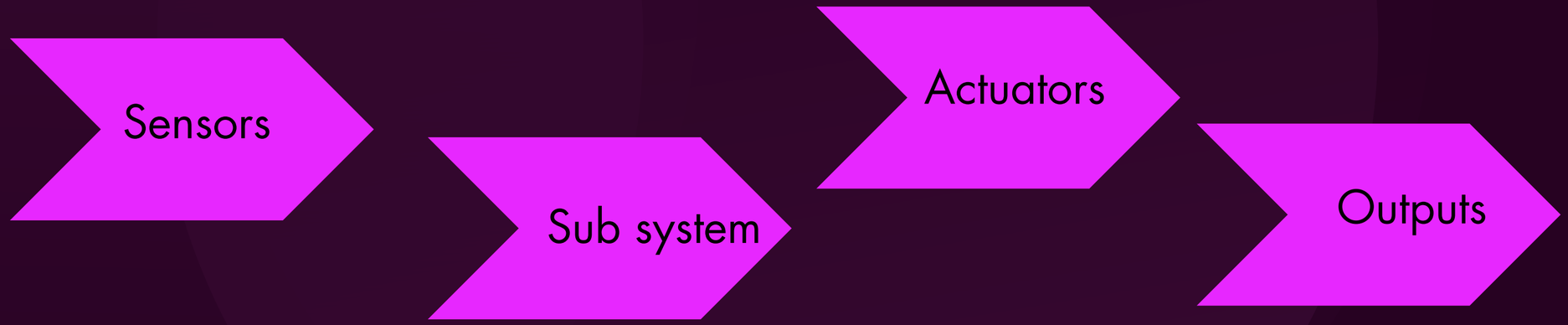
Solution:

Observation Tensor Design



Feedback Loops Model





Mathematical Model

01 System State

$$S_t = \{P_t^i, d_t^i, L_t^i, B\}$$

$P_t^i = [x_t^i, y_t^i]$ Agent Position

$d_t^i = [d_{x,t}^i, d_{y,t}^i]$ Direction

$L_t^i = \{(x_k, y_k, \tau_k)\}_{k=1}^{n_i}$ Light Trail

02 Agent´s actions

$$A_t^i = (m_t^i)$$

$[0, -1]$ Up

$[-1, 0]$ Left

$[0, 1]$ Down

$[1, 0]$ Right

$d_{t+1}^i = m_t^i$ Next direction

03 Environment Dynamism

$$L_i = L_{i-1} \cup \{x_t, y_t\}$$

Rewards System

01 +100 if they would get by eliminating the opponent with their own light trail

02 -20 if the agent die

03 -0.1 for each step taken without dying.

04 +200 to the team on the winning

Tools used

01

PyTorch:
flexibility and native support for RLlib which simplifies multi-agent training

PyTorch was chosen because it is a powerful and flexible tool, ideal for multi-agent reinforcement learning projects.

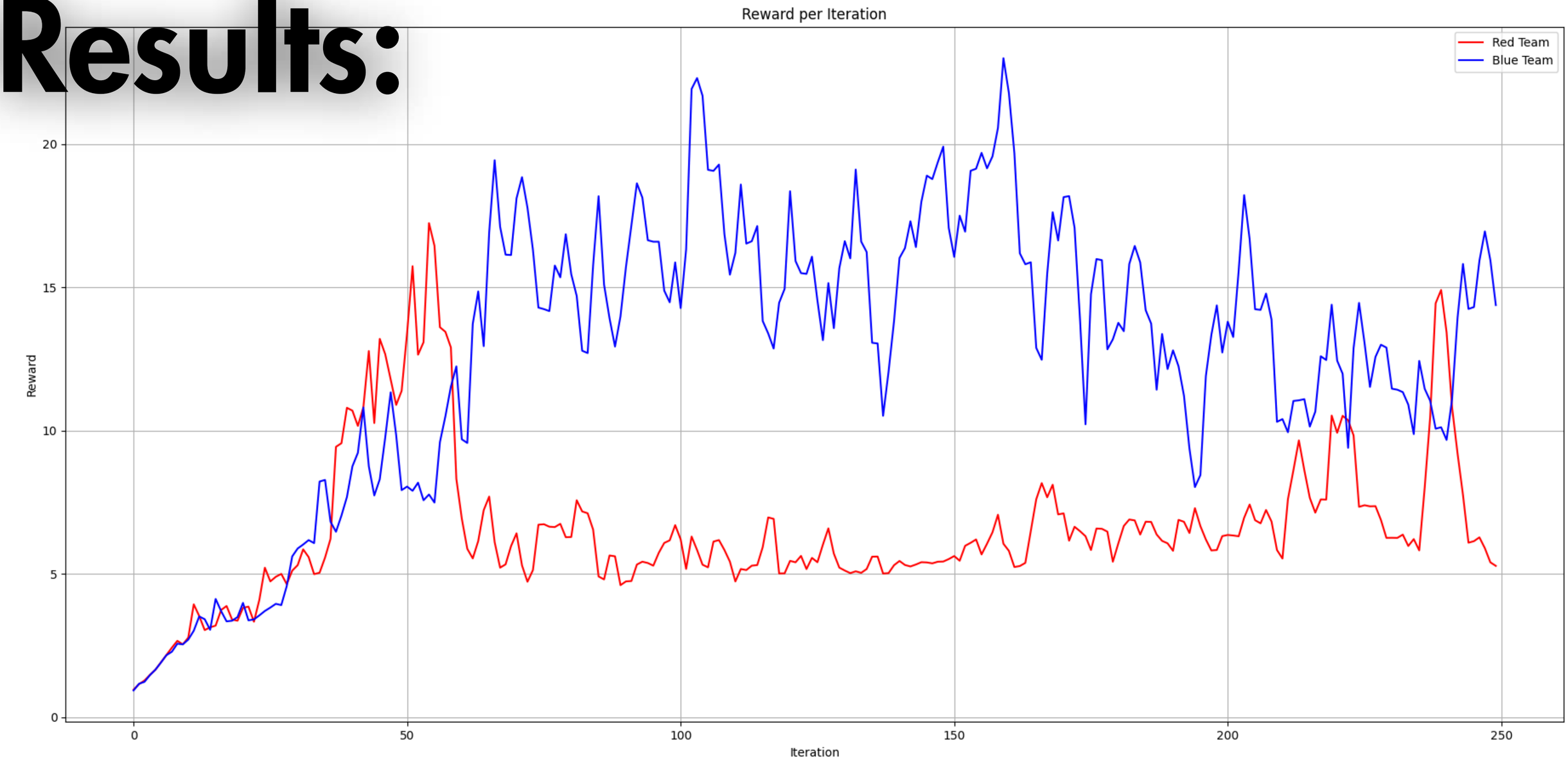
02

RLlib: scalable library for reinforcement learning, supporting multi-agent training and custom environments. We use PPO for its stability and efficiency in dynamic competitive tasks.

03

Petting Zoo:
Offers environment for the agent training in a multi-agent context.

Results:


Reward per Iteration
Red Team
Blue Team
Reward
Iteration
0
5
10
15
20
0
50
100
150
200
250