Workshop No.3

Buitrago, Erick
Student ID: 20221020072

Jiménez, Juan
Student ID: 20221020087

Universidad Distrital Francisco José de Caldas
June 2025

Contents

Introduction

This document presents the updated design of a multi-agent learning system inspired by the Tron game, with a focus on vision-based interaction, cybernetic feedback integration, and communication between agents. Key modifications include replacing the full-map observation tensor with a directional vision cone, implementing a new reward system to encourage aggressive play, and introducing message-passing protocols for cooperative decision-making. The learning algorithm selected is MAPPO, which supports both competitive and cooperative behaviors under partial observability. It is also proposed a simple machine learning model applying its corresponding features and labels. These enhancements aim to create more realistic agent behavior, improve coordination, and increase overall the system for testing advanced reinforcement learning strategies.

1. Previous Workshop Corrections

In this section it is found a few corrections of the previous workshop. The information of this section is because it is relevant to other aspects to the actual content of this workshop or because changes were made.

1.1. Functional Specifications

Sensors: The sensor is a vision-based model, which is able to gather information such as obstacles, opponent's positions as well as light trails and the time they take to disappear. This information will be used for processing and decision making. (A further explanation is given in the "Cybernetic Feedback Integration" section)

Actuators: The actuators correspond to the input keys used to control the agent's movement and to activate or deactivate the light trail, i.e., to leave the barrier behind or not.

Rewards: The reward system changed to promote an aggressive play style. The rewards the agents have consist of a macro-reward, to complete the main goal, and other sub-goals to achieve it. The main objective of the agents would be to win the game by having an aggressive play style, so the reward they would get by eliminating the opponent with their own light trail would be much more significant, they will get +100 by doing it. They would get a lower reward if the opponent crashes on its own with a border or their own trail, they will get +50.

To achieve the main goal there are other sub-goals, e.g., surviving, it was taken before as the main objective but it would not achieve the wanted competition among the agents, so it will be a little penalty -0.1 that the agent will get for each step taken without dying. (The reason to change it is explained in the "Cybernetic Feedback Integration" section).

As one must compensate the good behavior on the agent, it must also be penalized the unwanted one, the agent will have -20 by dying with any obstacle present.

1.2. Mathematical Model

The game environment is defined as a discrete-time dynamical system on a two-dimensional grid of size $M \times N$, where $M = 34$ and $N = 20$ represent the width and height of the full map, respectively. The set of valid positions (or playable cells) is denoted by $\mathcal{P} \subseteq \mathbb{Z}^2$, excluding the outermost border. Thus, the effective playable area is 32×18 cells.

At each discrete time step $t \in \mathbb{N}$, the system state is represented as:

$$\mathcal{S}_t = \left\{P_t^1, P_t^2, d_t^1, d_t^2, L_t^1, L_t^2, B\right\}$$

Where:

- $P_t^i = [x_t^i, y_t^i] \in \mathcal{P}$ is the position of agent i at time t.
- $d_t^i = [d_{x,t}^i, d_{y,t}^i] \in \{-1, 0, 1\}^2$ is the direction vector of agent i at time t.
- $L_t^i = \{(x_k, y_k, \tau_k)\}_{k=1}^{n_i}$ is the set of light trail positions left by agent i, each with remaining lifetime τ_k (measured in seconds).
- B represents the static borders of the map.

The evolution of the system is governed by the transition function:

$$\mathcal{S}_{t+1} = T(\mathcal{S}_t, A_t^1, A_t^2)$$

Where $A_t^i = (m_t^i, \ell_t^i)$ denotes the action of agent i, consisting of:

- $m_t^i \in \{[0,-1],[0,1],[-1,0],[1,0]\}$ for movement (Up, Down, Left, Right).
- $\ell_t^i \in \{0,1\}$ for the light trail toggle (Off, On).

The updated position is calculated as:

$$P_{t+1}^i = P_t^i + m_t^i$$

and the updated direction is:

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

If $\ell_t^i = 1$, the agent leaves a new trail segment at P_t^i with an initial duration $\tau = 10$ seconds. Trails decay over time and are removed once their lifetime is over.

Collisions with walls, other agents, or light trails result in terminal states for the affected agents. Each agent's reward r_t^i is computed based on the following:

- $+100$ if agent i eliminates an opponent using its trail.
- $+50$ if the opponent dies on its own.
- $+0.1$ per step survived.
- -20 for dying.

The full system evolves according to both internal dynamics (agent movement and trail decay) and external inputs (agent actions). This formulation sets the foundation for modeling, training, and analyzing agent behavior using reinforcement learning techniques.

2. Machine Learning Implementation

2.1. Algorithms and Frameworks

Before talking about the algorithm that will be used, it will be necessary first to talk about the historical data that can be useful to take into account. Historical data that agents will know before hand are the time it takes the light trail to fade away (10 seconds), their speed (7 cells per second), the field/map size (34x20 squares including exterior borders, which would be 32x18 playable squares) and the average duration of a game.

Taking into account the different machine learning models taught in class, the one that fits more is Reinforcement Learning (RL). By using historical data they will be more aware of the space they are taking part off, however, for the training it is more suitable to use RL on-policy, i.e., based on the agent's transitions it will explore and learn.

As for the data they will know during their training they will be aware of their position in the map, their current direction and the objects visible in their range (further explanation in Cybernetic Feedback Integration).

Still, a simple machine learning algorithm is proposed to implement off-policy training. The selected features include the remaining time before a light trail disappears and its

distance from the agent. The target label is a movement direction, aiming to determine whether the agent can safely move straight through the trail or should instead turn to avoid it. It can also be used to infer an opponent's position based on the trail they leave behind.

Taking into account that the system is a Mixed Cooperative-Competitive Setting, it was searched an algorithm able to fulfill the needs. The one that fits more is the Multi-Agent Proximal Policy Optimization (MAPPO), this allows the cooperation of members of the same team while competing against their opponents as part of their reward system.

2.2. Cybernetic Feedback Integration

In the earlier stages of development, agents perceived their environment through a 9-channel observation tensor, which encoded spatial information such as walls, player positions, light trails, and movement directions. However, with the redesign of the game's visual system, this tensor was deprecated. A more realistic visual sensor is now proposed, based on limited direction.

Each agent is now equipped with a triangular vision cone resembling a flashlight beam, oriented in the direction the agent is facing. Any object obstructing the cone such as walls, teammates, or opponents will block visibility beyond that point, simulating occlusion. However, light trails do not obstruct vision, maintaining their role as hazards rather than obstacles.

To implement this vision model, the map was redesigned from pixel-based to a discrete grid of cells. This allows the vision cone to be expressed as a triangle of grid positions. Additionally, the total number of grid cells is expected to increase (i.e., smaller cell size), which enables fine-tuning of the vision cone's resolution and depth.

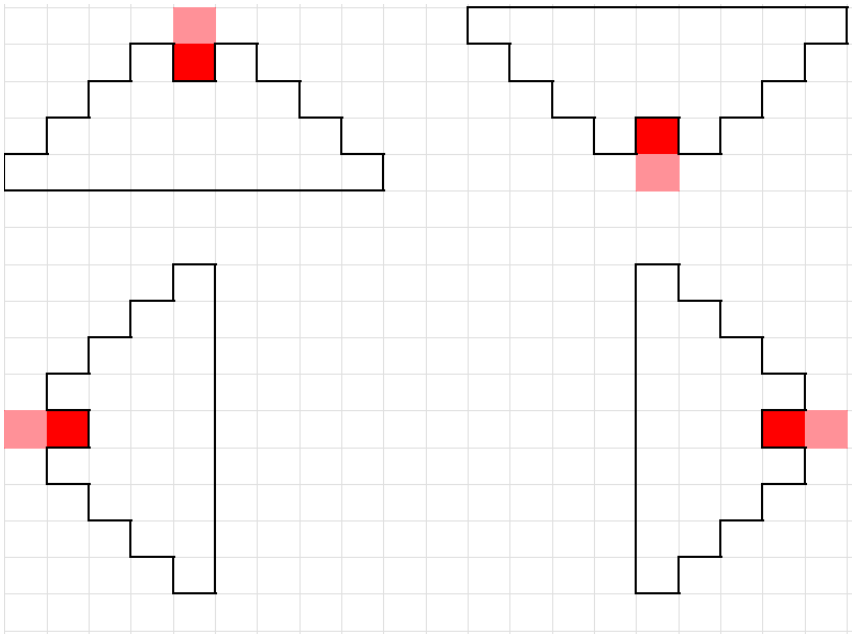

Figure 1: 4 different cases of the triangular field of view based on the agent's direction.

The agent's perception of the environment relies on a directional cone-shaped vision sensor. This sensor is represented as a triangular field of visible cells oriented in the agent's current direction. Figure 1 illustrates the four possible configurations based on the agent's orientation:

- **Upward (↑)**: The vision cone expands northward, with a wide base at the top and the vertex at the agent. It allows mid-range forward detection with wide horizontal coverage.
- **Downward (↓)**: The cone is projected southward with the same shape inverted. Useful when retreating or facing threats from below.
- **Leftward (←)**: The cone opens to the west, projecting the field to the left. It covers multiple lateral cells while maintaining forward depth.
- **Rightward (→)**: The cone expands eastward, mirroring the leftward case. It helps anticipate collisions or opponents from that direction.

This sensor has a configurable range (currently 4 cells deep and 9 wide) and can be adjusted according to the map's size. Solid objects such as walls or agents will cause occlusion, preventing visibility of any cells behind them. In contrast, light trails do not block the field of vision.

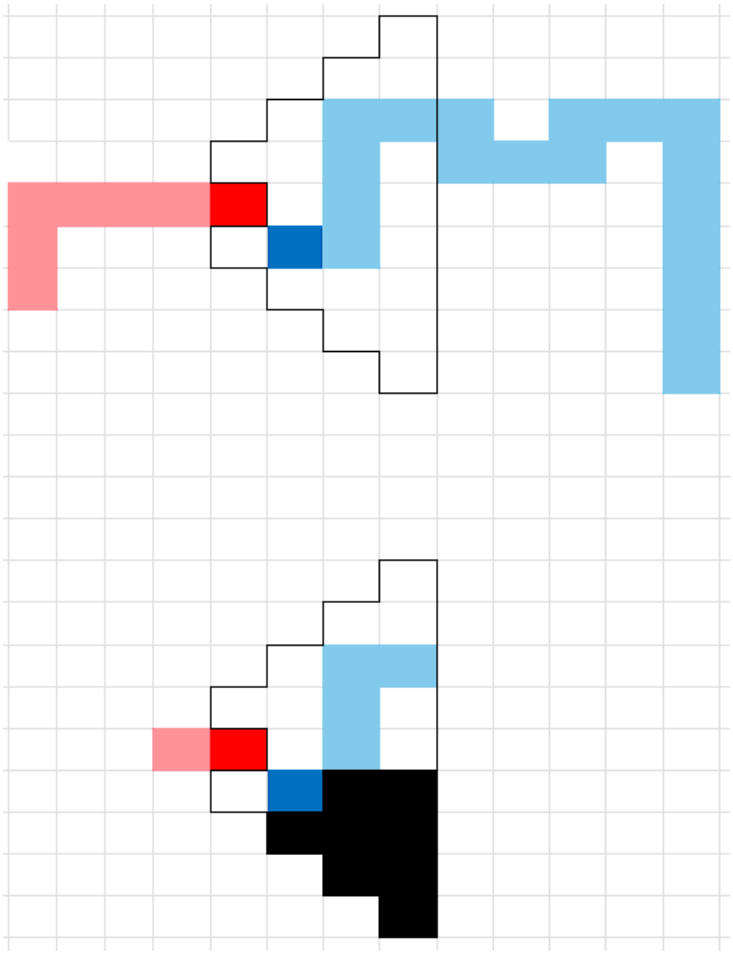

Figure 2: Case study: Vision Triangle. Above: The game's perspective. Below: The red agent's perspective.

The figure 2, illustrates the behavior of the directional vision sensor with occlusion implemented in the red agent. The top half shows a full view of the environment (spectator perspective), where the red agent is facing forward with its cone shaped vision area, and

the blue agent is visible along with its light trail. This represents the real state of the environment with no visibility constraints.

The lower half shows how the red agent perceives the world using its vision sensor. The black cells represent areas that are occluded by solid objects, in this case, the blue agent. The visible cells within the triangle are what the red agent can "see," including the blue trail. Light trails are considered transparent and do not block vision, but they are visible to them. This distinction makes the perception model more realistic.

This vision system is based on a triangle of grid cells projected in the direction the agent is facing, introducing limited and directional perception. It fosters more complex and strategic behavior, such as hiding, predicting collisions, or setting up ambushes.

The sensory data from the vision cone can be stored either as:

- A `set` of coordinate tuples, which is efficient for quick comparison and prevents duplicates.
- A 2D matrix representing visible cells, suitable for consumption by AI agents or neural networks (similar to a single channel tensor).

On the feedback side, a granular reward system has been implemented with a focus on promoting aggressive yet strategic behavior. The reward system includes:

- High positive rewards for eliminating opponents using the light trail.
- Smaller penalties for crashing into walls or allied trails.
- A small penalty per step to encourage aggressive behavior. (It was decided to change it to a penalty instead of a reward, as it causes the agents not competing, i.e., when facing an opponent turning around, now their goal would be to stop the game sooner eliminating their opponents).
- Control based feedback for the trail activation mechanism, which includes a success probability, adding uncertainty to the agent's decision making.

This cybernetic loop based on sensory input, reward driven adaptation, and directional interaction with the environment aims to emulate autonomous, emergent behavior typical of reinforcement learning systems.

3. Agent Testing and Evaluation

3.1. Experimental setup

A series of progressively challenging training scenarios will be deployed to evaluate and shape the agents' strategic behavior. Initially, the environment will run without light trails enabled, allowing agents to focus on basic navigation and avoidance learning. A symmetric 2-vs-2 setup will be used, with the blue team spawning on the right half of the board and the red team on the left, both with initial movement directed toward the opponent. This setup encourages immediate interaction and early learning of core behaviors such as avoiding walls, breaking loops, and staying active.

Subsequently, light trail activation will be enabled, allowing each agent to decide when to toggle their trail on or off. This action includes a stochastic component: a 70% success

rate and a 30% chance of failure, introducing uncertainty into the agent's control loop and encouraging adaptive planning.

Once foundational behavior has been established, initial conditions will be altered to test agent generalization. This will ensure that the agents are not overfitting to a single static configuration and can adapt to varying environments and scenarios. The following variations will be introduced during training and testing:

- Random spawn positions to encourage spatial adaptability.
- Starting near walls or between opponents to assess escape and survival strategies under "pressure".
- Variations in vision range and cone shape to test robustness of perception and decision-making.
- Placing opponents within the field of vision from the start to test initial reactions and aggression.
- A pool of four predefined maps will be used to introduce structural diversity. Each game will randomly select one of these maps, shown in Figure 3, to provide different obstacle layouts and environmental constraints. The black squares in each map indicate the standard spawn positions for the agents under normal starting conditions.

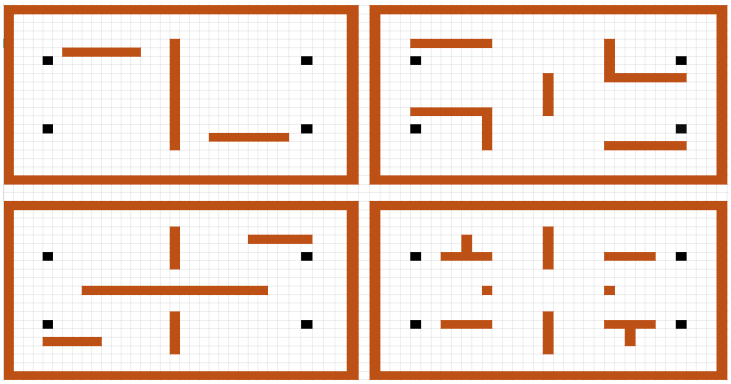

Figure 3: Four different maps

Finally, additional experiments will modify trail visibility and activation reliability to assess how agents adjust their strategies in response to perceptual and control variability.

3.2. Performance Metrics

Although training episodes have not yet been executed, a series of learning objectives and performance metrics have been defined to evaluate agent behavior within the environment. Among them, the most critical is the metric related to defeating opponents, as the main goal of the system is to encourage aggressive play.

For each episode, the following values will be recorded:

- K: Number of kills caused by the agent's trail.
- D: Number of deaths (collisions against walls or trails).

- T: The total episode duration measured in time steps.

We define an Aggressiveness Index (AI) as follows:

$$AI = \frac{K}{T} \times \frac{K}{K + D + 1}$$

This metric favors agents that eliminate opponents quickly while minimizing self-destructive behavior. The index will be used as a primary success criterion: if, over multiple episodes, the average aggressiveness index surpasses all other tracked metrics (such as survival time or total reward), we can conclude that the training is effectively promoting the desired aggressive learning behavior.

Other complementary metrics include:

- **Cumulative reward per episode:** measures the total reward accumulated by the agent during one episode, providing a global performance score.
- **Average number of steps survived:** Calculated as the mean duration (in time steps) each agent remains active before being eliminated, over a batch of training episodes.
- **Trail activations to kills ratio:** Evaluates how efficiently the agent uses its light trail by comparing the number of trail activations to the number of resulting kills.
- **Win/loss count over training windows:** Tracks the number of victories and defeats in predefined training intervals, offering insights into consistency and learning stability.

These metrics will be visualized and analyzed to assess the convergence, effectiveness, and consistency of the learned policies.

4. Multi-Agent Extension

4.1. Communication Protocols

The main communication protocol that is proposed to make the game more "realistic", is to include **message passing**, this would allow planning strategies between members of the same team. Message passing consists of sending some specific information with a signal or a vector to its teammate. The information that is crucial to be sent is the opponent's position and directions, as well as any light trail they detect with their visual sensor.

Another valuable application of message passing involves the detection of dynamic threats. When an agent perceives a potential collision or a sudden change in the opponent's trajectory within its cone of vision, it can immediately broadcast an alert to its teammate. This enables a cooperative reaction, such as evasive maneuvers or even setting up a trap by adjusting their relative positions. Sharing this type of situational awareness allows the agents to behave as a coordinated unit rather than as isolated individuals.

Moreover, message passing can be extended to include strategic intents, rather than just states. For example, an agent could inform its teammate of its next planned move or a specific area it intends to control, helping to avoid overlapping paths and enhancing area coverage. This strategy and coordination could be useful, where occupying key

regions of the map or blocking enemy paths becomes a shared goal. By allowing agents to dynamically update their strategies through minimal communication, the team's overall performance and adaptability can be significantly improved.

4.2. Cooperative or Competitive Interactions

The proposed system incorporates cybernetic feedback loops to allow autonomous agents to learn and adapt their behavior through constant interaction with the dynamic game environment. Instead of operating with a global view of the board, agents rely on a directional vision sensor with occlusion.

This directional sensor acts as the perceptual input of the cybernetic loop. The visual information collected is converted into an internal representation of the environment (for example, a set of coordinates or a matrix), which is then evaluated by the agent's policy. Based on this partial observation, the agent selects an action: turning, toggling the light trail, or continuing straight. Once the action is executed, the environment updates and returns a new observation and a numerical reward, thus closing the feedback loop. These rewards are carefully designed to promote strategic and aggressive behaviors (positive for eliminating opponents, negative for collisions, passiveness, or inefficient trail activation). This cycle enables each agent to iteratively refine its policy and optimize performance under partial observability and competition.

In the multi-agent setting, these cybernetic loops function both independently and cooperatively. Each agent operates based on local observations and personal reward signals, leading to distributed decision-making without central control. However, through mechanisms like message passing, individual feedback loops become interconnected. For instance, if an agent detects an enemy within its vision cone and transmits that information, its teammate can incorporate this data into its own loop, adapting its actions accordingly. This inter-agent communication enriches the learning process, enabling the emergence of coordinated strategies such as blocking opponents or territory control. Ultimately, the distributed architecture ensures that each agent reacts autonomously while contributing to a shared team objective, creating a system of dynamically coupled feedback loops that adapt to a constantly evolving game environment.