Technical Report: Analysis and Design of an Adaptive Autonomous Predator-Prey Agent System

Juan Diego Grajales Castillo (20221020128)
Nicolás Castro Rivera (20221020055)

May 15, 2025

Contents

List of Figures

List of Tables

Abstract

This technical report details the analysis and conceptual design phase of a multi-agent system simulating predator-prey dynamics in a discrete 2D environment. Following Workshop 2 specifications, we present rigorous mathematical models, adaptive feedback schemes, and proposed reinforcement learning algorithms. The document includes: (i) functional specifications of sensors and actuators, (ii) nonlinear motion equations with Jacobian stability analysis, (iii) feedback loop diagrams, and (iv) literature-grounded algorithmic recommendations. Experimental results are omitted as they belong to subsequent phases. The methodology follows an iterative-incremental approach, prioritizing theoretical stability and replicability.

1 Introduction

1.1 Context and Motivation

Predator-prey interaction simulation enables the study of emergent phenomena in complex adaptive systems. This project integrates concepts from cybernetics and evolutionary game theory to model autonomous learning agents. The current version (Workshop 2) incorporates feedback from Workshop 1, focusing on:

- Simplifying reward functions to reduce learning noise
- Formalizing mathematical motion equations and stability analysis
- Integrating feedback loop diagrams to clarify decision-making

1.2 Objectives

- Design a stochastic discrete environment with reactive/proactive agents
- Establish theoretical framework for convergence and chaotic behavior analysis
- Propose scalable algorithmic pipeline for future implementations

2 Literature Review

Table 1 summarizes key theoretical foundations. Deep Reinforcement Learning (DRL) has proven effective in partially observable environments (Mnih et al.,), while Jacobian linearization remains standard for discrete dynamical systems (Strogatz,). Studies like (Hu et al.,) validate sparse reward functions for accelerated convergence.

Table 1: Comparative approaches in multi-agent simulations

Parameter	**Classical Approach**	**Proposed**
State space	Continuous (Differential Eqs)	Discrete (Cellular Automata)
Reward function	Dense (e.g., Euclidean distance)	Sparse (+10/-5 per capture)
Update	Continuous time	Discrete steps ($\Delta t = 1$)

3 Methodology

3.1 Iterative-Incremental Design

We adopt Scrum framework with conceptual sprints focusing on:

- Rapid prototyping with basic Q-Learning
- Mathematical formalization and stability analysis

3.2 Functional Specifications

3.2.1 Perception and Action

- **Sensors**: 5×5 cell radius (Max detection: $d_{\max} \approx 7.07$)
  - Predator: Detects prey (± 0.5 cell error), walls, traps
  - Prey: Detects predator with $\tau = 2$ step delay

- **Actuators**: 4-direction movement (N, S, E, W) with constraints:

$$v_p(t) \in \{(0,1),(0,-1),(1,0),(-1,0)\}, \quad \|v_p(t)\| = 1 \tag{1}$$

3.2.2 Reward Functions

Sparse structure to prevent local over-optimization:

- Predator: $R_p(t) = \begin{cases} +10 & \text{successful capture} \\ -0.1 & \text{idle step} \end{cases}$
- Prey: $R_h(t) = \begin{cases} -5 & \text{captured} \\ +0.1 & \text{successful evasion} \end{cases}$

3.3 Mathematical Modeling

3.3.1 Motion Equations

Discrete update with logistic adaptation:

$$\pi_p(t+1) = \pi_p(t) + r_p(t) \cdot v_p(t) \tag{2}$$

$$r_p(t) = \nu_{\max} + \frac{\alpha}{1 + e^{-kd(t)}} \quad (\alpha = 2.0, k = 0.5) \tag{3}$$

3.3.2 Local Stability

Jacobian linearization of map F around equilibria (π_p^*, π_h^*):

$$J = \begin{bmatrix} \frac{\partial F_p}{\partial \pi_p} & \frac{\partial F_p}{\partial \pi_h} \\ \frac{\partial F_h}{\partial \pi_p} & \frac{\partial F_h}{\partial \pi_h} \end{bmatrix}, \quad \text{Stability if } \max(|\lambda_i(J)|) < 1 \tag{4}$$

3.4 Proposed Software Architecture

Key implementation components:

- **Environment**: GridWorld with OpenAI Gym wrappers
- **Agents**: Python classes with ϵ-greedy policies
- **API**: REST interface for real-time monitoring

4 Suggested Algorithms

Table 2 compares proposed methods. Actor-Critic prefers continuous spaces, while DQN handles partial observations.

Table 2: Reinforcement learning algorithm comparison

Algorithm	**Strength**	**Limitation**
Q-Learning	Simplicity	Table explosion
DQN	Generalization	Instability
SARSA	Online policy	Exploration bias
Actor-Critic	Stability	Sensitive hyperparameters

5 Conclusions

The theoretical analysis establishes foundations for a stable, replicable system. Sparse reward structures and logistic speed adaptation are key contributions. Next steps include: (i) DQN implementation with experience replay, (ii) numerical Jacobian eigenvalue validation, and (iii) Gaussian noise introduction for robustness.

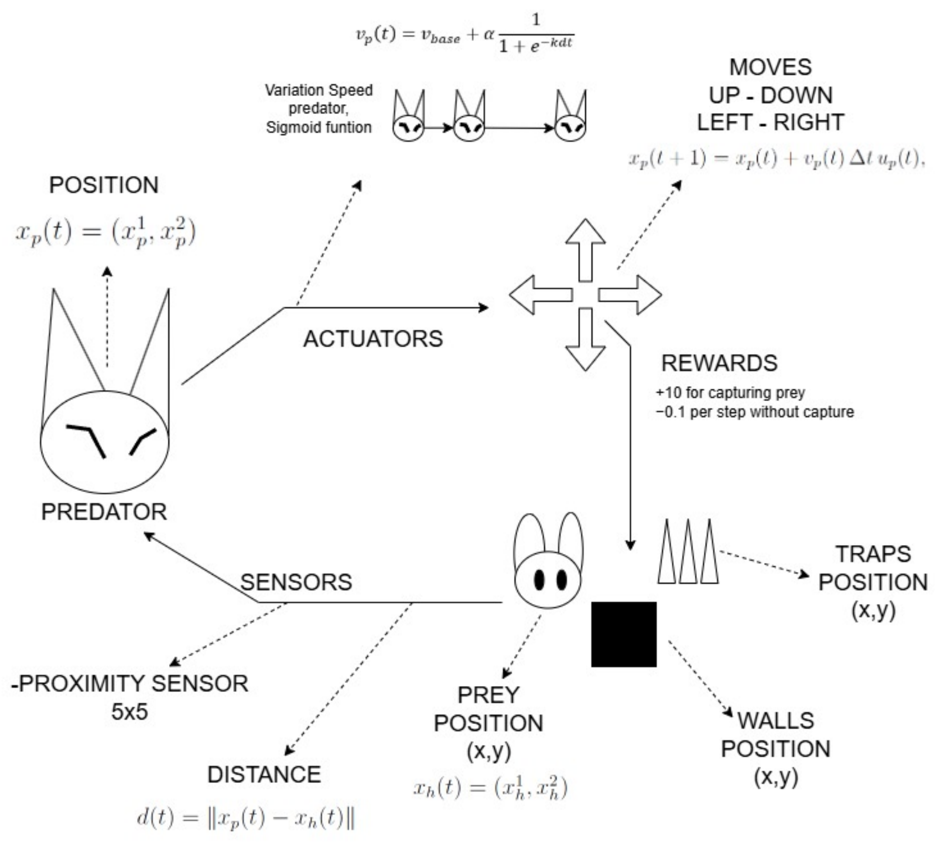
References

Hu, H., et al. (2020). Sparse rewards in reinforcement learning: A survey. *IEEE Transactions on Neural Networks*, *31*, 1–15.

Mnih, V., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

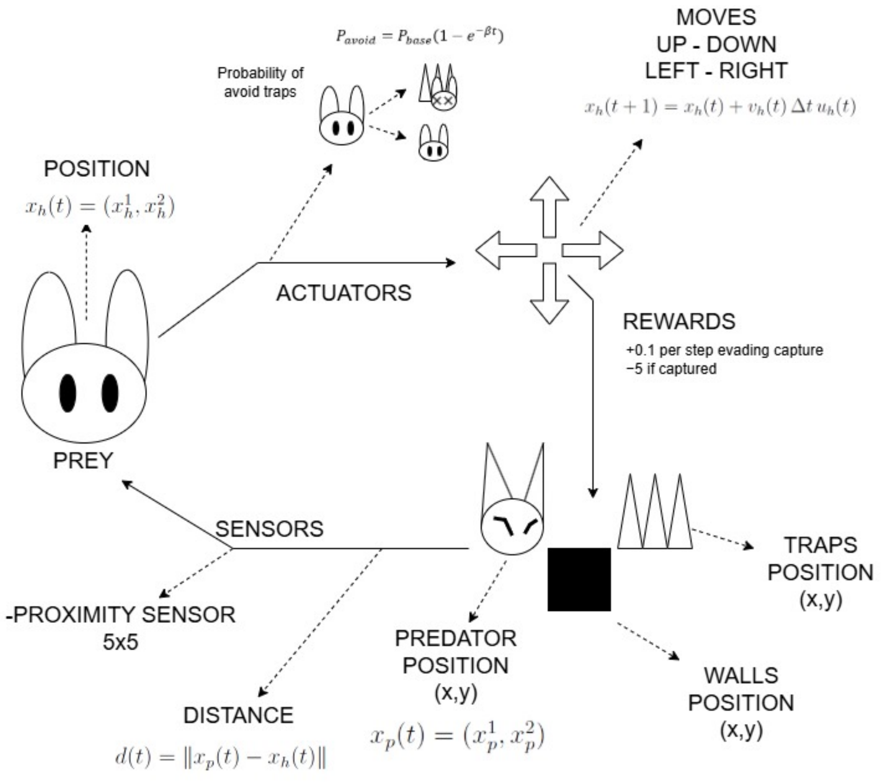
Strogatz, S. H. (2018). *Nonlinear dynamics and chaos.* CRC Press.

A Feedback Loop Diagrams



$v_p(t) = v_{base} + \alpha \frac{1}{1+e^{-kdt}}$
Variation Speed predator, Sigmoid funtion
MOVES UP - DOWN LEFT - RIGHT
$x_p(t+1) = x_p(t) + v_p(t)\,\Delta t\, u_p(t),$
POSITION
$x_p(t) = (x_p^1, x_p^2)$
ACTUATORS
REWARDS
+10 for capturing prey
−0.1 per step without capture
PREDATOR
SENSORS
TRAPS POSITION (x,y)
-PROXIMITY SENSOR 5x5
PREY POSITION (x,y)
$x_h(t) = (x_h^1, x_h^2)$
WALLS POSITION (x,y)
DISTANCE
$d(t) = \|x_p(t) - x_h(t)\|$

Figure 1: Detailed diagram: Predator feedback loop (Workshop 2)



$P_{avoid} = P_{base}(1 - e^{-\beta t})$
Probability of avoid traps
MOVES
UP - DOWN
LEFT - RIGHT
$x_h(t+1) = x_h(t) + v_h(t)\,\Delta t\,u_h(t)$
POSITION
$x_h(t) = (x_h^1, x_h^2)$
ACTUATORS
REWARDS
+0.1 per step evading capture
−5 if captured
PREY
SENSORS
TRAPS POSITION (x,y)
-PROXIMITY SENSOR 5x5
PREDATOR POSITION (x,y)
$x_p(t) = (x_p^1, x_p^2)$
WALLS POSITION (x,y)
DISTANCE
$d(t) = \|x_p(t) - x_h(t)\|$

Figure 2: Detailed diagram: Prey feedback loop (Workshop 2)