WORKSHOP #3 – MACHINE LEARNING & CYBERNETICS

Johan Sebastián Gutiérrez Pérez - 20211020098

Francisco Jose de Caldas District University

System Sciences Foundations

Eng. Carlos Andrés Sierra

June 12th, 2025

Introduction

This third and final workshop in the Systems Sciences course marks a pivotal stage in the development of an autonomous adaptive agent. Building upon the system design principles established in Workshop 1 and the dynamic system modeling explored in Workshop 2, this phase focuses on the practical implementation of machine learning and cybernetic control. The goal is to transform theoretical constructs into a functional agent capable of navigating a simulated urban environment, making decisions autonomously, and adapting to dynamic conditions.

To guide the agent's learning process, we employ **Q-learning**, a model-free reinforcement learning algorithm particularly suited to discrete environments. Its simplicity and interpretability make it ideal for our scenario, where the agent operates in a grid-based city and learns from interactions within a bounded and well-defined state space. Through trial and error, the agent updates a Q-table that represents the expected utility of taking specific actions in given states, allowing it to gradually discover optimal navigation policies.

The agent is equipped with a local sensor that perceives a fixed-size neighborhood — conceptualized as a 3x3 grid— enabling it to detect nearby obstacles, delivery targets, and potential threats. A cybernetic feedback loop is designed to link these sensor readings with the Q-learning process, shaping the agent's reward function to reinforce desirable behaviors such as efficient delivery and collision avoidance.

In addition to standard delivery tasks, a stochastic element is introduced into the environment in the form of a moving bird. This dynamic obstacle compels the agent to respond not only to static conditions but also to unpredictable environmental changes, further testing its adaptability and decision-making capabilities.

This document presents the architecture of the implemented learning system, details the integration of cybernetic principles into the learning and control mechanisms, and provides an analysis of the agent's performance across several evaluation metrics. By completing this workshop, the agent achieves a functional level of autonomy, capable of learning from experience and adjusting its behavior in real time.

1. Environment and Agent Design

1.1. Environment Overview

The simulated environment is represented as a discrete two-dimensional grid, abstracting a simplified urban layout consisting of streets, buildings, and designated delivery zones. Each cell on the grid holds a specific semantic value—either traversable or blocked—thus defining the operational boundaries for the autonomous drone. In addition to static obstacles, a dynamic element has been introduced: a bird that randomly moves throughout the city. This moving obstacle introduces an additional layer of uncertainty and complexity, compelling the drone to adjust its strategy in real time and avoid potential collisions.

1.2. Sensor Model

The drone is equipped with a cybernetic sensing mechanism based on a local 3×3 neighborhood window centered on its current position. This vision model provides the agent with partial observability of its surroundings, including nearby obstacles (e.g., buildings), dynamic threats (e.g., the bird), and the direction of the delivery target. Each cell in the 3×3 grid is encoded into a simplified state vector, allowing the agent to generalize across different

environmental patterns and maintain a balance between spatial awareness and computational tractability.

1.3. Action Space

The drone's action space consists of five discrete movements: move up, down, left, right, or stay in place. These actions allow the agent to navigate the grid environment with flexibility, enabling both reactive and proactive behavior based on sensor feedback and the agent's learned policy. All actions are constrained by environmental boundaries and the position of static and dynamic obstacles.

1.4. State Representation

The agent’s state is represented by a flattened encoding of the 3×3 observation grid, capturing information such as obstacle presence, goal proximity, and dynamic threats. This state vector serves as the input for the Q-learning algorithm, facilitating learning from localized environmental feedback. The design ensures compatibility with tabular Q-learning by maintaining a manageable number of possible state permutations, while still encoding sufficient information for decision-making.

1.5. Cybernetic Feedback Loops

Sensor data is continuously mapped to the agent's internal state through a feedback control loop, aligning with core principles of cybernetics. When the agent faces changes in the environment—such as the sudden appearance of the bird or the blocking of a previously

accessible route—these changes are detected through the vision system, translated into updated state representations, and then used to adjust the agent's behavior accordingly. This loop enables the system to self-regulate and adapt in real time, maintaining its delivery objective despite environmental fluctuations.

1.6. Learning Algorithm: Q-Learning

The reinforcement learning engine driving the agent's adaptability is based on the classical Q-learning algorithm. At each time step, the agent updates a Q-table based on the observed state, chosen action, resulting reward, and subsequent state. The learning process is governed by key parameters: the learning rate (α), discount factor (γ), and an ε-greedy exploration strategy to balance exploration and exploitation. The reward function is tightly coupled with the sensory feedback loop, offering positive reinforcement for reaching the delivery target, penalizing collisions with obstacles or the bird, and discouraging inefficient paths.

2. Machine Learning Implementation

2.1. Algorithm Selection

For the learning module of this project, the Q-learning algorithm was selected due to its simplicity, interpretability, and suitability for discrete environments with manageable state-action spaces. Given the local 3×3 observation grid and the finite action set, Q-learning provides a tractable solution without requiring function approximation or neural networks, making it ideal for rapid experimentation and debugging.

Additionally, Q-learning allows fine-grained control over exploration strategies, learning rates, and discounting behavior, all of which are critical for tuning the agent's performance in dynamic, partially observable environments such as the one in this simulation.

2.2. **Implementation Details**

The agent maintains a Q-table indexed by discretized states (encoded from the 3×3 vision grid) and actions. The update rule used is the classical Bellman equation:

$$Q(s_{t,}a_t) < - Q(s_{t,}a_t) + \alpha[r_t + \gamma max_a\, Q(s_{t+1}, a) - \, Q(s_{t,}a_t)]$$

Where:

- α: Learning rate.
- γ: Discount factor.
- r_t: Immediate reward received after taking action a_t.
- $s_{t,}s_{t+1}$: Current and next states

In simple terms:

- The agent updates how good it thinks an action is ($Q(s_{t,}a_t)$) based on the reward r it receives and the best option it sees in the next stage ($max_a\, Q(s_{t+1}, a)$).
- α controls how fast it learns.
- γ controls how much it cares about future rewards.

2.3. **Reward Structure**

The reward function integrates cybernetic principles by tying sensor feedback directly to learning signals:

Scenario	Reward
Reaching the delivery goal	+100
Colliding with a building	-50
Colliding with the bird	-100
Each time step (movement)	-1
Staying still	-2

This structure encourages efficient navigation, penalizes dangerous or redundant behavior, and reinforces the desired system behavior in alignment with self-regulating goals.

2.4. Reward Function and Cybernetic Feedback Loop

To guide the learning of the drone, we define a reward function that reflects how well the agent is performing within its environment. This function translates the drone's interactions with its surroundings—captured through its 4×4 local sensor grid—into numerical feedback. This feedback becomes the basis for learning and adaptation.

The reward scheme is structured as follows:

- **+100** for successfully delivering a package to the correct location.
- **-100** for colliding with a building, the boundaries of the city, or the bird (a dynamic obstacle).
- **-10** for taking unnecessary steps, encouraging efficiency.
- **+1** for moving closer to the target delivery location.
- **-1** for moving further away from the target.

This system creates a cybernetic feedback loop, the drone uses local sensor inputs to perceive its current state, selects an action based on its learned Q-values, receives a reward, and adjusts its behavior accordingly. This loop enables the agent to self-regulate and improve over time, adapting to both static (buildings) and dynamic (bird) elements in the environment.

By consistently applying this feedback mechanism, the agent becomes more resilient and adaptive, key qualities in cybernetic systems.

3. Agent Testing and Evaluation

3.1. Experimental Setup

To validate the learning and adaptability of the autonomous drone, a comprehensive experimental setup will be designed. The environment simulates a city grid where the drone must navigate toward delivery targets while avoiding collisions with buildings and a dynamic obstacle (a bird) that moves unpredictably across the map. The scenarios will include:

- **Randomized initial conditions**: The positions of both the drone and the delivery targets will vary across episodes to promote generalization.
- **Dynamic obstacle variability**: The bird will follow different random paths in each simulation run, controlled by distinct random seeds.
- **Environment modifications**: The size of the grid and the density of static obstacles will be adjusted in selected test cases to assess the agent's robustness under changing conditions.

- **Reproducibility control**: Fixed random seeds will be used in some test runs to enable consistent benchmarking and comparison between different agent versions or hyperparameter settings.

The combination of these scenarios ensures both adaptability to new situations and consistent performance under controlled conditions.

3.2. **Performance Metrics**

To evaluate the learning progress and effectiveness of the autonomous agent, several quantitative metrics will be collected and analyzed throughout the training and testing phases:

- **Episode Reward**: The cumulative reward obtained by the agent in each episode will serve as a primary indicator of learning progress. This metric will help assess whether the agent consistently achieves its objectives, such as delivering packages successfully while avoiding collisions.

- **Learning Curve**: The average episode reward will be plotted across training episodes to visualize the agent's performance trend over time. This curve will provide insights into how quickly the agent improves its policy and whether learning plateaus or degrades at any stage.

- **Convergence Speed**: The number of episodes required for the agent to achieve stable, near-optimal performance will be measured. This metric will help compare the efficiency of different hyperparameter settings or variations in the environment.

- **Collision Rate**: The frequency with which the agent collides with obstacles (including buildings and the dynamic bird) will be tracked as an additional safety

metric. A decreasing collision rate over time will indicate effective learning of safe navigation behaviors.

- **Delivery Success Rate**: The proportion of episodes where the agent successfully delivers its package without collisions will be recorded to provide a direct measure of task completion effectiveness.

All these metrics will be logged and visualized using appropriate tools (e.g., Matplotlib or built-in logging utilities from Stable-Baselines3) to support quantitative analysis and ensure that the agent meets the predefined performance objectives.

References

Wiener, N. (1948). *Cybernetics: Or control and communication in the animal and the machine*. MIT Press.

Watkins, C. J. C. H., & Dayan, P. (1992). **Q-learning**. *Machine Learning*, *8*(3-4), 279–292. https://doi.org/10.1023/A:1022676722315

Stable-Baselines3 Contributors. (2024). *Stable-Baselines3 documentation*. https://stable-baselines3.readthedocs.io/

Gymnasium Contributors. (2024). *Gymnasium documentation*. https://gymnasium.farama.org/

PyTorch Team. (2024). *PyTorch documentation*. https://pytorch.org/docs/stable/index.html