Applying Q-Learning to Herding Behavior: A Single-Agent Approach in NetLogo

Bernardo A. Santos[1], **Fernando Santos**[1]

[1]Departamento de Engenharia de Software
Universidade do Estado de Santa Catarina (UDESC)
Ibirama – SC – Brazil

bernardo.santos@edu.udesc.br, fernando.santos@edu.udesc.br

Abstract. *This paper presents an agent-based simulation in NetLogo to evaluate Q-learning within a classic shepherding problem across two distinct operational configurations. The simulation evaluates a static baseline featuring fixed agent initialization and a stationary target, alongside a randomized dynamic scenario characterized by stochastic spawning and an actively moving sheep. Utilizing a low-complexity discrete state space based on relative distances and directional vectors, the dog agent learns to guide the target toward a designated gate via a reward mechanism, completely independent of pre-programmed heuristics. Rather than reporting isolated final metrics, the core analysis evaluates the behavioral stabilization and learning trajectories of the agent under varying environmental complexities. The experimental results demonstrate a policy optimization in both configurations, tracking how the system minimizes operational steps and stabilizes reward retention over extended training timelines.*

1. Introduction

Computer simulations are essential tools for researching complex phenomena and supporting decision-making processes when dealing with numerous variables. However, developing systems of such high complexity for analysis is far from trivial. To address this challenge effectively, we rely on Agent-Based Model Simulation (ABMS). ABMS shifts the analytical focus onto the individual (the agent), capturing the complex, emergent behaviors that arise purely from their localized interactions with both the environment and other individuals around them [Macal and North 2010].

To operate efficiently in dynamic environments, autonomous agents must leverage artificial intelligence techniques to adapt and optimize their choices over time. Reinforcement Learning (RL) provides a robust toolkit for this adaptation [Watkins and Dayan 1992], enabling agents to discover the most effective behavioral strategies by directly learning from their own successes and mistakes.

The task of sheep herding represents a classical and challenging benchmark problem in spatial agent interaction and autonomous control. The fundamental complexity of shepherding lies in the indirect manipulation of a non-cooperative, reactive agent (the sheep) by a driving agent (the herding dog) [Azuma et al. 2012] and [Strömbom et al. 2014]. To successfully guide the sheep toward a designated target gate, the herding agent cannot rely on conventional, static point-to-point pathfinding. Instead, it must execute complex geometric movement logic based on the real-time relative positioning between itself, the sheep, and the goal. The dog must navigate to an optimal

repulsion vector directly behind the sheep, aligning its approach angle with the target. Any miscalculation in operational distance can either fail to trigger the sheep's perception response or, conversely, cause it to panic and bolt in an undesirable direction, generating non-linear and chaotic environmental feedback.

To evaluate how an autonomous agent can systematically discover these intricate spatial rules without pre-programmed heuristics, this paper models a foundational scenario consisting of a single dog agent tasked with herding a single sheep toward a target gate [Strömbom et al. 2014]. Using an agent-based simulation built in NetLogo, we isolate the spatial mechanics of the perception-zone behavior to analyze the adaptability and viability of using the Q-learning algorithm [Watkins and Dayan 1992]. The agent's behavioral evolution and operational efficiency are empirically evaluated using reinforcement learning metrics, focusing on the progression of training episodes, the minimization of total steps required to complete the herding task, and the optimization and stabilization of accumulated rewards. Results demonstrate a clear optimization in the agent's trajectory efficiency, characterized by a reduction in total operational steps alongside the simultaneous stabilization of both episodic rewards and the success rate.

2. Background

2.1. Agent-Based Modeling and Simulation

Agent-Based Model Simulation (ABMS) is a computational paradigm that uses individual, decentralized agents to reproduce and study complex systems [Macal and North 2010]. In this framework, the simulation is built from autonomous agents placed inside a shared environment that they can actively perceive and modify through their actions. These agents serve a fundamental role: they possess their own independent decision-making rules, allowing them to interact dynamically with both the physical environment and other individuals around them. ABMS is effective because, much like the real world, it seamlessly captures localized complexity, emergent behaviors, and a true sense of spatial awareness.

While agent-based simulations can be built from scratch using generic programming languages, specific platforms provide dedicated tools that drastically simplify development and reduce model complexity [Railsback et al. 2006]. NetLogo is a specialized modeling framework designed for simulating natural and social phenomena. It provides a high-level programming language with native commands for navigation, environment configuration, and direct agent interaction. Utilizing a Cartesian coordinate system, NetLogo's grid architecture serves as the perfect foundation for designing spatial movement, geometric logic, and reinforcement learning tasks.

2.2. Reinforcement Learning and Q-Learning

Reinforcement Learning (RL) is a branch of machine learning where an autonomous agent learns to maximize its performance based on direct feedback (rewards and punishments) received while exploring an unfamiliar environment. Instead of having its actions and values pre-programmed by a developer, the agent discovers which choices lead to the highest cumulative reward through continuous, hands-on interaction with the world around it [Watkins and Dayan 1992]. In contemporary research, this paradigm has proven effective for complex spatial coordination tasks, underpinning recent advancements in both

curriculum-based swarm shepherding [Hussein et al. 2022] and reactive robotic herding along dynamic paths [Van Havermaet et al. 2024].

One of the most well-known and widely used types of reinforcement learning is Q-Learning [Watkins and Dayan 1992], serving as a core approach that still underpins recent advancements in autonomous robotic herding [Van Havermaet et al. 2024]. This paradigm operates by computing expected cumulative rewards for discrete state-action pairs, formally quantified as Q-values. Each Q-value, represented mathematically as Q(s,a), estimates the highest possible cumulative reward that the agent can expect to receive in the future if it starts in a specific state (s), executes a chosen action (a), and continues to follow an optimal decision-making strategy thereafter. The agent constantly updates and maintains a Q-table, which acts as its memory matrix storing the quality scores for every possible state-action combination. As the agent interacts with the environment, it transitions into a new state (s') and receives an immediate reward (r). It then uses this real-time feedback to update the corresponding value in the Q-table, calculating its course correction according to the Bellman Equation 1:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{1}$$

This learning process relies on the following key parameters:

- **Learning Rate (α):** A value between 0 and 1 that determines how much the agent values new information compared to what it has already stored in memory. A high learning rate makes the agent adapt quickly to new situations, while a low learning rate ensures it retains stable, long-term strategies.
- **Discount Factor (γ):** A value between 0 and 1 that balances whether the agent prioritizes immediate rewards or long-term success. A lower value keeps the agent focused entirely on short-term gains, while a higher value teaches it to value future rewards, such as the ultimate goal of guiding the sheep through the gate.
- **Action Selection Policy (ε):** A strategy used to balance exploration and exploitation. The agent behaves greedily (choosing the best known action based on its Q-value) nearly all the time. However, with a probability of ε, it selects a completely random action, forcing the agent to explore new possibilities and discover better paths.

3. Methodology and Environment Modeling

To successfully apply Q-learning to a herding simulation, the environment must be structured and built using low-complexity components. These foundational development structures function by clearly defining the States, Actions, and Rewards, which together serve as the framework to guide the dog through its learning process [Watkins and Dayan 1992]. Accordingly, the following subsections systematically detail how these pillars are formalized into concrete environmental states, discrete agent actions and the reward shaping mechanism. Additionally, the underlying spatial dynamics and behavioral rules governing this environment are grounded in the classic multi-agent shepherding frameworks proposed by [Lien et al. 2004] and [Strömbom et al. 2014].

3.1. States

The core of the simulation developed in this paper is an agent interaction between a dog and a sheep inside an environment. The ultimate objective is for the dog to learn how to guide the sheep through a designated target gate.

State is simply a snapshot of the environment at a specific moment. It shows where the dog, the sheep, and the gate are in relation to each other. Instead of using raw map coordinates, the state uses geometric values that link the dog's position directly to its goal. The dog looks at this state to decide its next move, which changes the environment and keeps the simulation moving forward.

The behavior of the sheep is defined by two distinct movement states based on the proximity of the dog relative to a perception zone. When the dog breaches this boundary, the sheep takes evasive action and moves in the exact opposite direction to escape [Strömbom et al. 2014]. Conversely, if the dog remains outside this perception zone, the sheep executes a random move in any direction. Because the sheep is a dynamic target, the dog cannot rely on a simple straight-line pursuit; instead, it must learn to continuously evaluate the surrounding environment to achieve successful herding. To accomplish this without relying on complex, continuous coordinate tracking, the continuous 2D environment is abstracted into a low-complexity discrete state space. This state space is strictly defined by three fundamental geometric dimensions calculated relative to the dog, the sheep, and the target gate: distance, angle, and alignment.

The first dimension is the relative distance between the dog and the sheep, which is discretized into two primary states: "near" and "far" (Figure 1(A)). The "near" state triggers when the dog penetrates the sheep's drifting zone, while the "far" state represents positions outside this boundary. The second dimension captures the relative approach angle of the dog with respect to the sheep. The entire 360-degree field surrounding the sheep is divided into 8 distinct sectors of 45 degrees each (Figure 1(B)), allowing the agent to identify its relative angular positioning. The third dimension evaluates the alignment vector, which defines the geometric positioning of the dog relative to the sheep-to-gate axis. This alignment is mapped into 4 possible discrete states (Figure 1(C)), representing whether the dog is perfectly positioned behind the sheep, offset to either side, or improperly aligned in front of the sheep. By combining these three dimensions, the algorithm constructs a finite matrix of combinations (2x8x4), allowing the reinforcement learning agent to accurately categorize its spatial context and select the optimal command to guide the sheep.

3.2. Actions

Actions represent the choices the agent makes to move and navigate the simulation. By taking actions such as moving forward or turning the dog directly changes its position, modifies the environment, and creates an entirely new state [Watkins and Dayan 1992]. Within the model, these discrete actions serve as the driving force that dictates the agent's direction and behavior, allowing it to actively manipulate the simulation to achieve its goal.

Following the foundational spatial tracking and relative positioning principles established in classic shepherding frameworks [Lien et al. 2004] and

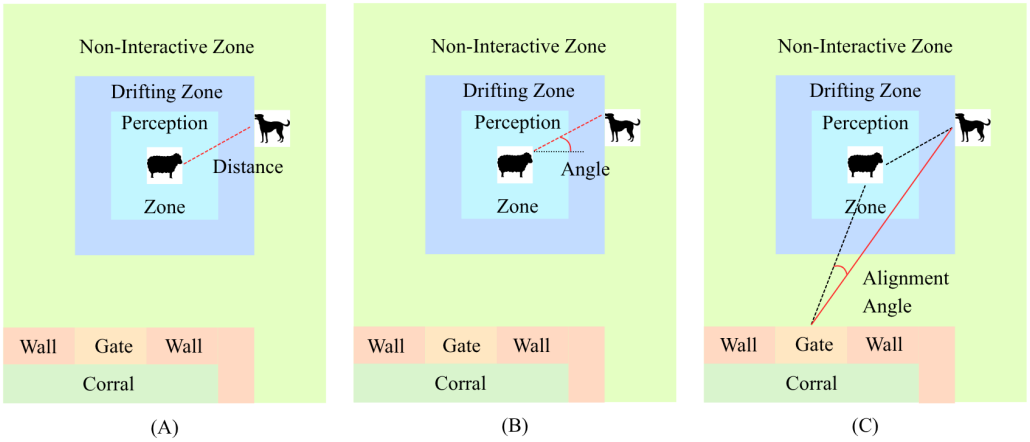

Figure 1. Three spatial parameters that specify the states

[Strömbom et al. 2014], this model formally maps the continuous spatial coordinates of the NetLogo grid into a structured, low-complexity state representation. Instead of passing raw, high-complexity coordinate pairs to the Q-learning algorithm, these continuous spatial relationships are calculated at each simulation tick and discretized to determine both the active state matrix and its permitted action paths.

The spatial proximity (d) between the herding dog's coordinate position and the sheep's position is tracked using standard Euclidean distance. Within this spatial framework, the agent's policy selects from a discrete action space consisting of basic locomotion (Move Forward, Turn Left, and Turn Right), as well as specialized herding maneuvers: Orbit Trigger, which allows the dog to surround the sheep while remaining safely outside its immediate perception threshold; and Tangent Movement, which enables the dog to move along a circular trajectory to adjust its positioning without triggering the perception zone. This scalar value directly dictates the operational mode of the herding agent by partitioning the environment into three behavioral zones, as illustrated in Figure 2.

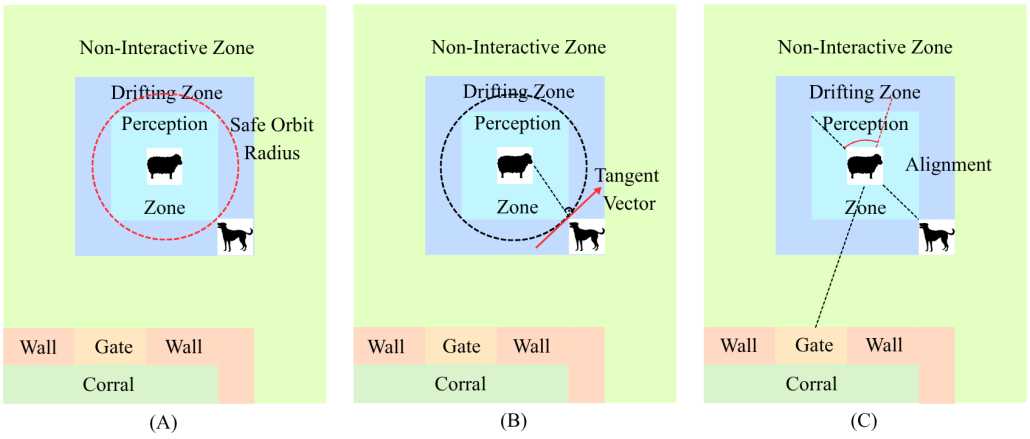

Figure 2. Actions Representation

- **Zone A (Non-Interactive Zone):** Triggered when the Euclidean distance is large. In this zone, the dog is outside the sheep's perception threshold. The state is

categorized as “far”, mapping directly to an acceleration policy where the agent prioritizes basic locomotion actions (Move Forward, Turn Left, and Turn Right) to rapidly intercept the target.

- **Zone B (Drifting Zone):** Defines a tactical window where the dog is close to the sheep but has not yet triggered its flight response. Entering this zone opens the state matrix paths for the Orbit Trigger and Tangent Movement actions. The agent calculates this zone specifically to execute orbital corrections around the sheep without scattering it.
- **Zone C (Perception Zone):** Establishes the critical proximity threshold where the sheep detects the dog and flees, fixed at a Euclidean distance of 1. When the state registers as “near”, any directional Move Forward action by the dog creates immediate environmental feedback, forcing the sheep to move away along a repulsive vector.

Complementing this proximity framework, the agent tracks its directional orientation relative to the target to guarantee accurate trajectory adjustments. As shown in Figure 3(A), the relative angle (θ_{ds}) represents the global heading required to look directly at the sheep from the dog’s current position, calculated via the two-argument arctangent function, shown in Equation 2.

$$\theta_{ds} = \text{atan2}(y_s - y_d, ; x_s - x_d) \tag{2}$$

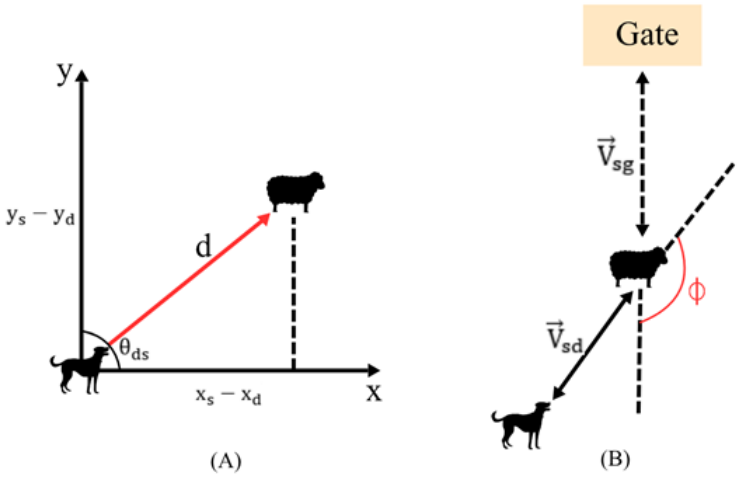

Figure 3. Mathematical framework of the state variables

This parameter determines the absolute vector relative to the global environment grid. By continually evaluating the difference between its current internal heading and θ_{ds}, the algorithm discretizes the angular context into eight distinct directional sectors. This calculation maps directly to the rotational actions, informing the dog of the required adjustment needed to execute a Turn Left or Turn Right action and face the sheep accurately, thereby preventing chaotic spatial deviations during approach maneuvers.

Ultimately, successful directional steering toward the goal is governed by assessing the trilateral alignment angle (Φ) between both agents and the final objective. As

illustrated in Figure 3(B), this metric measures the inner angle formed between the sheep-to-gate vector ($\vec{V}sg$) and the sheep-to-dog vector ($\vec{V}sd$), formalized using the geometric dot product 3:

$$\Phi = \arccos\left(\frac{\vec{V}sg \cdot \vec{V}sd}{|\vec{V}sg| \cdot |\vec{V}sd|}\right) \tag{3}$$

The alignment variable, represented by Φ, measures the directional accuracy of the dog's position relative to both the sheep and the target gate. Serving as the mathematical check that balances orbital adjustments with direct pushing, this metric abstracts continuous geometric space into discrete configurations. Consequently, the agent's spatial context is categorized into the following two operational states:

- Collinear Alignment ($\Phi \approx 180°$): The dog, sheep, and gate achieve near-perfect collinearity. This structural state validates the directional efficiency of a Move Forward push action, aligning closely with classical herding heuristics [Strömbom et al. 2014]. It signals to the policy that a forward movement will drive the sheep directly toward the target gate.
- Improper Alignment ($\Phi \neq 180°$): When Φ deviates significantly from $180°$, the state is classified as misaligned. In this situation, forward movement becomes undesirable because it may scatter the sheep toward the side boundaries. Consequently, the policy prioritizes Tangent Movement and Alignment Adjustment actions (Figure 3(B) and Figure 3(C)) until collinearity is restored.

3.3. Reward

Rewards quantify the value of every movement, scoring the immediate results of a specific state-action pairing. Positive rewards indicate progressive behavior that brings the agent closer to success, while negative rewards (penalties) discourage the agent from repeating mistakes. Ultimately, this reward function acts as the central engine of the learning mechanism, driving the dog to abandon random trial-and-error and develop focused herding behaviors.

The reward function begins with a baseline time penalty (rbase $= -0.1$) applied to every single step, providing continuous negative feedback that prevents aimless wandering and prioritizes path efficiency. This baseline is dynamically adjusted when the primary objective is achieved: once the sheep enters the designated goal area, the agent receives a massive, one-time positive reward of $+10$. This large value increments the success tracker (success-count) and acts as the mathematical anchor for the entire Q-matrix, validating the sequence of prior movements that led to a successful episode.

To prevent the agent from suffering from sparse-reward limitations, the environment provides immediate feedback based on the sheep's distance to the target gate (dgate) as follows:

- Progressive Alignment: If the sheep's new distance to the gate is less than its previous distance (new-dgate $<$ old-dgate), the dog is rewarded with $+4$ for successfully pushing the target forward.

- Regressive Alignment: If the sheep move further away from the gate (new-dgate $>$ old-dgate), a penalty of -0.5 is applied to discourage improper driving angles.
- Containment Threshold: If the dog moves too far from the sheep (new-ddog $>$ 7), a penalty of -0.5 is added. This teaches the sheepdog to maintain a tight operational radius and prevents the sheep from escaping its field of influence.

To prevent the dog from getting stuck or moving back and forth, the algorithm monitors its immediate position history. If the agent stays in the exact same spot (new-p $=$ old-p) or returns to its second-to-last position (new-p $=$ penultimate-p), a penalty of -1 is applied to the reward. This simple check eliminates repetitive loops and forces the dog to focus on active, progressive movements.

4. Results and Discussion

The simulation environment is constructed within a NetLogo coordinate grid measuring 6×10 patches, as shown in the environment layout (Figure 4), following established computational modeling recommendations. To evaluate the learning agent's adaptability, the framework implements two distinct experimental configurations: a static baseline and a dynamic scenario. In the static configuration, both the dog and the sheep spawn in the exact same fixed positions at the beginning of every training episode, and the sheep remains completely stationary. Conversely, the dynamic scenario applies a stochastic initialization strategy where both agents spawn randomly at any available position outside the corral, and the sheep actively moves throughout the environment as an autonomous, interacting agent [Strömbom et al. 2014]. Upon the completion of each episode, defined by the sheep successfully entering the gate, the simulation executes a complete environment reset, redistributing the agents according to the active scenario's rules. This dual-setup provides both a controlled reference and a randomized framework required to evaluate agent behavior in discrete spaces [Macal and North 2010]. The simulations are available online[1].

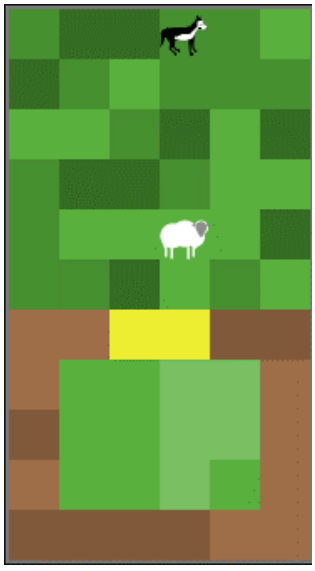

Figure 4. Visual Environment

[1] https://github.com/agentbasedsimulations/2026-wesaac-qlearning-herding-behavior-netlogo

To ensure efficient policy convergence, the algorithm's hyperparameters were selected using NetLogo's BehaviorSpace tool to conduct automated parameter sweeps. The variables, learning rate (α), exploration rate (ϵ), and discount factor (γ), were tested systematically across a range from 0.1 to 0.9 with increments of 0.1. Each configuration was evaluated over a fixed simulation duration of $100,000$ ticks, and the combination yielding the best overall performance outcomes was selected for the final model. Based on these empirical experiments, the optimal configuration was established with a Learning Rate (α) of 0.4 to balance new spatial rewards with past experiences, and a Discount Factor (γ) of 0.7 to align immediate maneuvers with long-term goal achievement. Action selection was governed by an Initial Epsilon (ε) of 0.5 to manage early exploration, which progressively minimizes over time via an Epsilon Decay Rate of 0.9994 as the agent masters the target geometric alignments.

To evaluate the long-term behavioral dynamics and stability of the reinforcement learning agent, Figure 5 tracks the cumulative number of completed episodes against the continuous advancement of global simulation time for both environmental configurations. In this visualization scheme, the x-axis represents the uninterrupted progression of simulation ticks from initialization to the final 500,000-tick threshold. Rather than resetting the timeline between individual trials, the global clock runs continuously; each time an episode is successfully completed by the sheep entering the gate, the event is recorded as an incremental step on the y-axis while the timeline advances forward. This cumulative logging style allows the reader to audit the agent's learning consistency, where a constant, upward slope indicates a reliable execution policy.

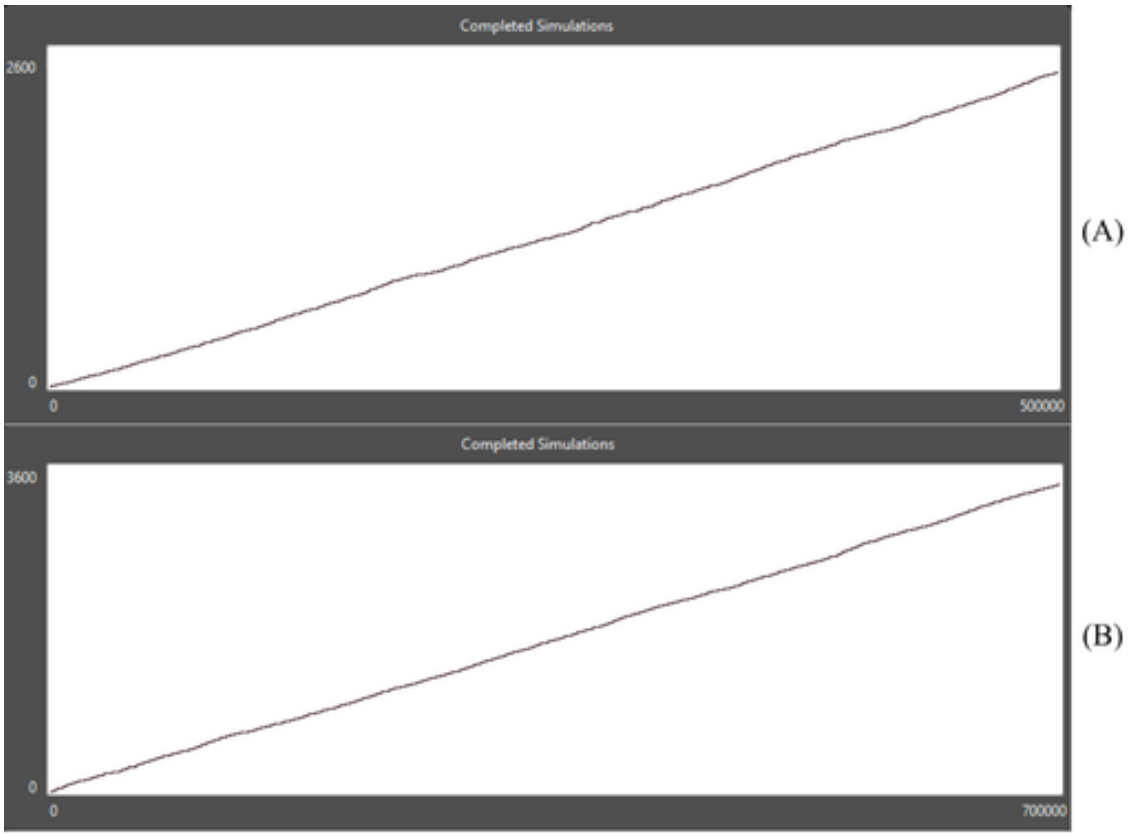

Figure 5. Completed Simulations Graph

For the static configuration Figure 5(B) presents the cumulative performance metrics for the static baseline configuration, where agent spawning locations are entirely fixed and the sheep remains completely stationary. Under these simplified, deterministic constraints, the 500,000-tick execution timeline resulted in a total of 3,600 successfully completed training episodes. This cumulative logging produces a straight linear trajectory, reflecting an stable and unvarying pace of task completion throughout the entire runtime. Because this environment completely eliminates the variables of target movement and initialization variance, the herding agent rapidly exploits the fixed state-action pairs, maintaining a predictable and flawless execution loop from start to finish without experiencing any behavioral fluctuations or optimization decay.

Conversely, Figure 5(A) characterized by random agent spawning positions and an actively moving, non-cooperative sheep the 500,000-tick execution timeline resulted in a total of 2,600 successfully completed training episodes. As demonstrated by the linear slope produced under these stochastic conditions, the agent maintains a consistent and uninterrupted rate of task completion across the entire runtime. This accumulation of successful runs confirms that despite the spatial randomness and shifting environmental states caused by the target's movement, the underlying Q-matrix successfully adapts and does not suffer from degenerative policy degradation or catastrophic forgetting over prolonged operational periods.

4.1. Learning Convergence, Rewards and Steps

To analyze the learning performance of the agent, Figure 6 presents a multi-panel layout organized by metric type and environmental configuration, applying a moving average calculated per 1,000 ticks. The top row, comprising Figures 6(A) and 6(B), monitors the total execution steps per episode, whereas the bottom row, consisting of Figures 6(C) and 6(D), tracks the cumulative episodic rewards. The subplots are aligned column-wise to correspond with each experimental setup: the left column—Figures 6(A) and 6(C)—illustrates the behavioral evolution of the dynamic simulation (Scenario A), while the right column—Figures 6(B) and 6(D)—isolates the performance metrics of the static baseline configuration (Scenario B). Across all four panels, a synchronized horizontal axis maps the continuous progression of the training timeline, enabling a direct evaluation of how operational efficiency and reward optimization correlate over time.

For the static baseline configuration (Scenario B)—characterized by fixed agent initialization and a stationary target—the relationship between the Steps Graph (Figure 6(B)) and the Reward Graph (Figure 6(D)) highlights a stabilization process across the 3,600-episode horizon. At the beginning of training, the system exhibits a brief and tightly contained exploration phase. Here, execution steps experience a sudden upward spike toward the 200-step maximum, while episodic rewards simultaneously undergo a quick oscillation near the 100-point limit as the agent maps the deterministic environment. Following this initial phase, both curves immediately transition into a rigid horizontal plateau. For the remainder of the 3,600 episodes, the execution steps lock into a uniform baseline, while the accumulated rewards settle into a consistent, unwavering line at nearly 100. This uniform flatline demonstrates that by eliminating the variables of target movement and spatial randomness, the agent masters and reproduces the single optimal herding trajectory within the allocated training window.

For the dynamic simulation (Scenario A)—characterized by random agent initial-

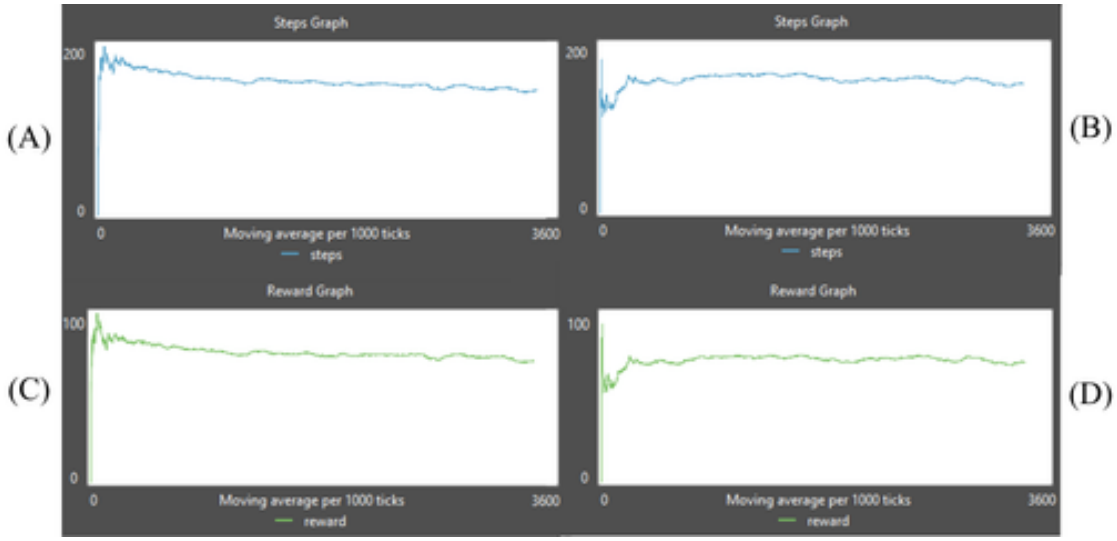

Figure 6. Steps and Reward graph for both simulations

ization and an actively moving target—the relationship between the Steps Graph (Figure 6(A)) and the Reward Graph (Figure 6(C)) highlights an adaptive learning process across the 2,600-episode horizon. At the beginning of training, the system exhibits a volatile exploration phase. Here, execution steps experience a sharp upward spike toward the 200-step maximum, while episodic rewards simultaneously undergo a pronounced oscillation near the 100-point limit as the agent executes unoptimized paths to map the environment. Following this initial phase, both curves smoothly transition into a stable plateau. For the remainder of the 2,600 episodes, the execution steps undergo a steady decline toward an efficient baseline, while the accumulated rewards settle into a consistent horizontal line. This synchronized progression demonstrates that despite the spatial unpredictability of a moving target, the agent successfully optimizes its herding policy within the allocated training window.

4.2. Limitations

It is important to acknowledge the limitations of the simulation, which are highlighted as directions for future research. No empirical comparison was conducted between the performance of the proposed Q-Learning agent and heuristic baselines or other reinforcement learning approaches, which prevents an objective assessment of its convergence speed and step efficiency. Likewise, the absence of statistical tests and standardized comparative metrics against competing methods limits the generalization of conclusions regarding the robustness of the learned policy. Furthermore, although this work presents two simulations static and dynamic, both operate with a discretized state space and a single herding agent, which restricts the scalability of the model to more complex multi-agent scenarios. These gaps are therefore recognized as opportunities for further investigation in subsequent work.

5. Conclusion and Future Work

This paper presented an agent-based simulation in NetLogo to evaluate a Q-learning algorithm applied to the task of sheep herding. By abstracting the continuous 2D environment into a discrete state space based on distance thresholds, approach angles, and

alignment vectors, the herding agent successfully mastered tactical pathfinding without relying on pre-programmed heuristics. The inclusion of a specialized hybrid Orbit Action provided the necessary kinematic constraints, allowing the dog to safely navigate around the sheep's perception zone without scattering. Empirical results confirmed policy convergence demonstrated by the stabilization of episode completion rate within fixed temporal windows. This single-agent configuration establishes a mathematically grounded baseline for spatial robotic shepherding under strict computational constraints [Lien et al. 2004].

Future research will focus on expanding the environment to include multi-agent flock dynamics. The next iteration of the model will introduce a flock of multiple sheep by integrating reactive Reynolds boids kinematics, which include cohesion, separation, and alignment forces. Consequently, the herding agent's state space will be adapted to calculate spatial vectors relative to the collective flock's center of mass and its trailing edge outliers, allowing the system to achieve more complex and scalable herding behaviors.

References

Azuma, S., Tabuchi, A., and Sugie, T. (2012). Modeling of sheepdog control. *Transactions of the Society of Instrument and Control Engineers*, 48(12):882–888.

Hussein, A., Petraki, E., Elsawah, S., and Abbass, H. A. (2022). Autonomous swarm shepherding using curriculum-based reinforcement learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, pages 633–641.

Lien, J.-M., Bayazit, O. B., Sowell, R. T., Rodriguez, S., and Amato, N. M. (2004). Shepherding behaviors with multiple shepherds. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.

Macal, C. M. and North, M. J. (2010). Tutorial on agent-based modelling and simulation. *Journal of Simulation*, 4(3):151–162.

Railsback, S. F., Lytinen, S. L., and Jackson, S. K. (2006). Agent-based simulation platforms: Review and development recommendations. *Simulation*, 82(9):609–623.

Strömbom, D., Mann, R. P., Wilson, A. M., Hailes, S., Morton, A. J., Sumpter, D. J. T., and King, A. J. (2014). Solving the shepherding problem: heuristics for herding autonomous, interacting agents. *Proceedings of the Royal Society B*, 281(1787):20140719.

Van Havermaet, S., Khaluf, Y., and Simoens, P. (2024). Reactive shepherding along a dynamic path. *Scientific Reports*, 14(1):14915.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3–4):279–292.