FRIteza

# Sheepdog-Driven Algorithm for Sheep Herd Transport

**Idora Ban, Simon Hehnen, Iva Idzojtic, and Lukas Pucher**

Collective behaviour course research seminar report

**December 7, 2024**

Iztok Lebar Bajec | izredni professor | mentor

**The project explores the sheepdog-driven algorithm for sheep herd transport, demonstrating how a single agent can control and guide a swarm from one location to another through simple local interactions. The objective is to replicate the algorithm and evaluate its effectiveness in simulating dynamic control, where the sheepdog influences the herd's movement using basic rules. The backward semi-circle reciprocation algorithm will be implemented, incorporating behaviors observed in real sheepdogs, such as circular path movement and decision-making based on the sheep's positions. The effectiveness of the algorithm in achieving coordinated herd transport will be assessed under varying conditions, including changes in herd size, obstacle placement, and sheepdog responsiveness. Ultimately, the goal is to enhance the algorithm's adaptability and explore its broader potential.**

Sheepdog-Driven algorithm | Swarm systems | Controlling agent

Animal behavior-inspired algorithms enable efficient, adaptive solutions to complex problems by mimicking the coordinated actions of animals, swarm behaviours, such as flocking or schooling. This project explores the Sheepdog-Driven Algorithm for Sheep Herd Transport by Liu et al., 2021 [1], that moves beyond traditional swarm systems, where collective behavior emerges naturally within a group, to address a reverse problem: how a single agent (a "sheepdog") can control and direct a swarm (a "herd of sheep") from one location to another. The objective is to replicate and evaluate this sheepdog-inspired approach, focusing on how a single controlling agent can dynamically influence the herd to accomplish transport tasks.

After implementing the base algorithm, our team will experiment with its parameters and performance under different conditions. This will include testing variations like obstacle placement, changes in herd size, or adjustments to sheepdog responsiveness. As we analyze the algorithm's effectiveness in guiding the sheep herd, we may consider developing new behaviors or optimizations to enhance adaptability, with the potential for further applications in fields like autonomous drone or robot control.

## Related Work

In recent years, numerous studies have advanced the understanding of collective behavior and herding control among autonomous agents. A foundational contribution by Reynolds (1987) [2] introduced the "boids" model, which used local rules to simulate natural group dynamics such as flocking and schooling. This work laid the groundwork for many modern algorithms in swarm robotics and serves as an essential reference for simulating decentralized animal behaviors. They suggested that future work could involve adding more realistic individual behaviors, such as hunger or fear, and integrating detailed animations like wing flapping with motion dynamics. These enhancements could make simulations more lifelike and versatile.

Building on this, Strömbom et al. (2014) [3] addressed the specific "shepherding problem," proposing heuristic methods for guiding groups of autonomous agents using a single "shepherd" agent—a framework that highlights key challenges and strategies relevant to single-agent control in herding tasks. Their approach emphasizes adaptive switching between collecting agents that are dispersed and driving the group when it is cohesive, a behavior closely resembling real-world herding events involving sheep and sheepdogs. Notably, their algorithm also highlights the importance of visual feedback, such as estimating spacing between agents, to maintain group cohesion and movement toward a target. This method demonstrates broader applicability, from robot-assisted herding and crowd control to tasks like environmental cleanup, making it a versatile solution for various collective behavior scenarios.

Furthermore, Bayazit, Lien, and Amato (2002) [4] proposed a global roadmap approach for managing group behavior in complex, obstacle-rich environments, underscoring strategies for effective navigation in challenging terrains. They demonstrated how embedding behavior rules into roadmaps and individual agents enables dynamic decision-making based on location and state. This approach mirrors the Sheepdog-Driven Algorithm, where the sheepdog dynamically adjusts its behavior based on the herd's position and proximity to obstacles. Both frameworks rely on balancing local interactions with global guidance, ensuring that individual agents respond adaptively

while maintaining overall group cohesion. The roadmap's ability to represent environmental constraints and influence movement aligns with the algorithm's goal of efficiently directing herds in diverse and complex scenarios.

Together, these studies form a comprehensive base for understanding swarm and herding behaviors, which informs the development of adaptive and efficient algorithms, such as the Sheepdog-Driven Algorithm, and supports a comparative analysis of its performance across various environments and control requirements.

## Methodology

The practical component of this project is divided into two primary tasks: implementing the simulation algorithm in Python and developing a visualization system in Unity. The methodology section outlines the approach taken to develop these components, along with the processes for testing, refining, and analyzing results.

**The Algorithm.** The algorithm defines the sheepdog's movement as a backward semicircular trajectory, allowing it to drive the sheep herd from behind [1]. The sheepdog operates within a two-dimensional plane, using an xx-yy coordinate system to track both its own location and that of each sheep. The sheepdog aims to guide the herd toward a designated target area by pushing them forward in a controlled manner.

Key aspects of the algorithm include the sheepdog's field of view, which restricts its awareness to only those sheep within its line of sight. Consequently, the sheepdog's response is dynamically adjusted based on the position and behavior of visible sheep, meaning it does not always have complete information about the entire herd.

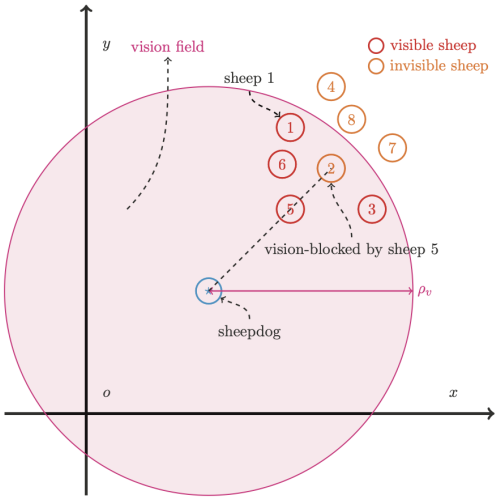


**Figure 1.** Example of dog's field of view.[1]

The control process of the sheepdog is divided into two phases:

1. Initialization: This phase involves setting key design parameters such as viewing angles, approach distances, and thresholds for direction adjustments. A time limit is also set for the operation.

2. Iteration: In the main phase, the algorithm evaluates the position of the sheep relative to the target and adjusts the sheepdog's movement accordingly. The sheepdog decides whether to move directly toward the herd or to take a detour based on the herd's overall position. When detouring, it aligns itself with the rightmost or leftmost visible sheep, adjusting its angle to efficiently steer the herd without unnecessary deviation from the target path.

The algorithm operates through a series of conditions to ensure energy efficiency and maintain herd cohesion, continuing until all sheep are guided to the target area or the time limit is reached.

The algorithm itself serves as the foundation for the Simulation phase, where its implementation will be tested and refined in a Python-based environment. Following the algorithm's execution, its performance will be evaluated in simulated conditions, with particular attention to the adaptability and effectiveness of the sheepdog's control in various test scenarios.

**Simulation.** In the simulation phase, the Sheepdog-Driven Algorithm will be implemented in Python to replicate the sheepdog-herd dynamics on a two-dimensional plane. The simulation aims to model the movements and interactions of both the sheepdog and the sheep, capturing the complex behaviors that arise as the sheepdog attempts to guide the herd to a target location. In the original paper, this was done with MATLAB.

1. Algorithm Implementation: The algorithm will be programmed to include the sheepdog's backward semi-circular motion, a key mechanism for maintaining control over the herd. This involves defining functions that represent the sheepdog's movements, perception field, and the reactive behaviors of individual sheep.

2. Parameter Tuning: To evaluate and optimize the algorithm's effectiveness, parameters such as the sheepdog's field of view, speed, and interaction range with sheep will be adjustable. This parameter tuning will allow us to observe how changes in these factors influence the sheepdog's ability to manage the herd under various scenarios.

3. Test Scenarios: Various test cases will be created to examine the algorithm's performance under different conditions, including:

    - Basic Movement: Verifying the algorithm's ability to guide the herd in open space.
    - Extended Dynamics: Introducing obstacles and outlier sheep to assess how the algorithm adapts to environmental complexity and group inconsistencies.
    - Different Agent Types: Modifying attributes such as the sheepdog's behavior, speed, or strategy to observe effects on control and cohesion within the herd.

4. This Python-based simulation will output path data for each sheep and the sheepdog, which will be saved and used as input for visualization in Unity.

**Visualisation.** The visualization component of this project will be developed in Unity to render the movement data generated from the Python simulation, providing an interactive and comprehensible format for analyzing the algorithm's behavior. This allows for a detailed observation of how effectively the sheepdog can guide the herd under various conditions. The visualization process begins with importing path data from the Python simulation into Unity . This data represents each agent's movements, enabling us to create animated paths that reflect the interactions between the sheepdog and sheep over time.

A virtual environment is then constructed in Unity to provide a realistic setting for the simulation. This environment includes basic elements such as ground planes, obstacles, and boundaries, mirroring the conditions modeled in the Python simulation to enhance visual coherence and aid in understanding how environmental factors affect the sheepdog's control over the herd.

Unity's visualization setup focuses on representing the dynamic interactions between the sheepdog and the herd. Various visualization techniques, including path trails, speed indicators, and field-of-view displays, are used to highlight important aspects of the behavior. This design provides insights into how the algorithm reacts to elements like obstacles, outliers, and different herd dynamics.

As the visualization is not rendered in real time, the animation will be recorded and played back in segments. This approach enables static analysis, where we can pause and closely examine specific interactions or adaptations within the algorithm. By allowing for playback and frame-by-frame inspection, the Unity-based visualization provides a valuable tool for evaluating the algorithm's effectiveness, adaptability, and overall performance in guiding the herd through diverse scenarios.

## Results

So far, we have implemented a simulation based on the sheepdog-driven algorithm for herd transport. The code models the interaction between the sheepdog and the sheep, with the dog guiding the herd towards a destination using local interaction rules. We have defined the movement dynamics for both the sheep and the sheepdog, including attraction, repulsion, and visibility checks. The simulation updates the positions of the agents in real-time.

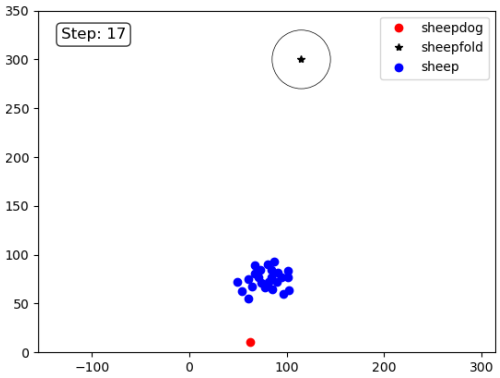


**Figure 2.** The starting phase of herding

Further refinement of the algorithm is underway to improve the movement and the mutual coordination.

## Discussion

After fully replicating the algorithm, the focus will shift to experimenting with different parameters to optimize its performance and achieve improved results. This includes exploring variations in herd size, sheepdog responsiveness, and environmental factors such as obstacle placement. Insights gained from these experiments will guide potential modifications or extensions to the algorithm, aiming to enhance its adaptability and effectiveness in diverse scenarios.

## Contributions

SH and IB did the implementation, II did the report, LP was helping with both.

## Bibliography

1. Liu Y et al. (2021) Sheepdog driven algorithm for sheep herd transport in *2021 40th Chinese Control Conference (CCC)*. pp. 5390–5395.
2. Reynolds CW (1987) Flocks, herds and schools: A distributed behavioral model in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87. (Association for Computing Machinery, New York, NY, USA), p. 25–34.
3. Strömbom D et al. (2014) Solving the shepherding problem: heuristics for herding autonomous, interacting agents. *Journal of the Royal Society Interface* 11.
4. Bayazit OB, Lien JM, Amato NM (2002) Better group behaviors in complex environments using global roadmaps in *Proceedings of the Eighth International Conference on Artificial Life*, ICAL 2003. (MIT Press, Cambridge, MA, USA), p. 362–370.