Sheepdog-Driven Algorithm for Sheep Herd Transport

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

Collective behaviour course research seminar report

January 10, 2025

Iztok Lebar Bajec | izredni professor | mentor

This project explores the Sheepdog-Driven Algorithm for Sheep Herd Transport, which demonstrates how a single agent can guide and control a swarm from one location to another using simple local interactions. The goal was to replicate and evaluate the algorithm's performance in directing a herd with a single controlling agent. A simulation and 3D visualization were developed to model and analyze the interactions between the sheepdog and the herd. The algorithm effectively demonstrated realistic behaviors, including attraction, repulsion, and visibility-based interactions, showcasing how local rules can influence collective movement. However, challenges in implementation prevented the herd from consistently reaching the target, highlighting areas where further refinement is needed.

Sheepdog-Driven algorithm | Swarm systems | Controlling agent

Introduction

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. The aim is to implement the algorithm from scratch, which allows for a deeper understanding of its mechanics and provides a more flexible, library-free approach to controlling the herd.

Related Work. Recent studies have advanced the understanding of collective behavior and herding control among autonomous agents. Reynolds (1987) [2] introduced the "boids" model, using local rules to simulate group dynamics like flocking and schooling. This work laid the foundation for modern swarm robotics algorithms and decentralized animal behavior simulations.

Strömbom et al. (2014) [3] addressed the "shepherding problem," proposing methods for guiding groups with a single "shepherd" agent. Their approach emphasizes adaptive switching between collecting dispersed agents and driving cohesive groups, resembling real-world herding. It also highlights the importance of visual feedback, such as agent spacing, to maintain cohesion and target movement. This method applies to robot-assisted herding, crowd control, and environmental cleanup.

Bayazit, Lien, and Amato (2002) [4] proposed a global roadmap approach for managing group behavior in complex environments. By embedding behavior rules into roadmaps, agents dynamically make decisions based on location and state. This mirrors the Sheepdog-Driven Algorithm, where the sheepdog adjusts its behavior based on the herd's position and obstacles, balancing local interactions with global guidance to efficiently direct herds in 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.

Methods

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 semi-circular 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.

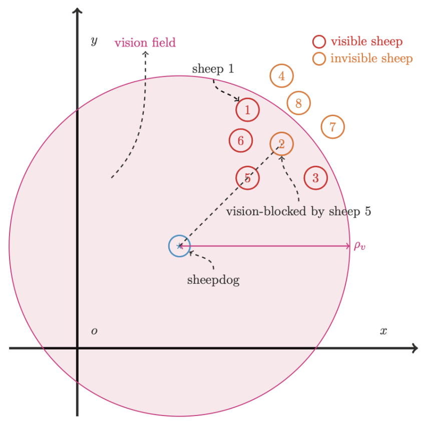

Figure 1. Example of dog's field of view.[1]

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.

Object-Oriented Approach. To simulate the behavior of the sheep and the dog, we employed an object-oriented programming approach using Python.

The core of the simulation is encapsulated within the Simulation class. All fixed parameters, such as the number of sheep, the destination position (sheepfold), the attraction radius, and the safety radius, are defined as class variables. Additionally, the current positions of the sheep and the dog are stored as NumPy arrays, ensuring efficient computation and manipulation.

Simulation. The simulation operates as a series of discrete steps. At the beginning of each step, the algorithm checks if convergence has been achieved—specifically, whether all sheep are within the destination circle. If this condition is met, the algorithm terminates. Otherwise, the simulation proceeds to calculate the updated positions of the dog and sheep.

1. Dog Movement: A velocity vector is computed for the dog. This velocity is scaled by a sampling period T and added to the previous position of the dog to determine its new position.

2. Sheep Movement: Each sheep's velocity is calculated based on its current state. This velocity is then transformed using a rotation matrix and added to its previous position.

After updating the positions, the step counter is incremented, and the old positions are stored. This historical data is essential for generating visualizations, as it provides the trajectory of both the sheep and the dog.

A simple visualization was implemented during development as a Python scatter plot. In this plot, the sheep and dog are represented by points of different colors, providing an intuitive view of their movements.

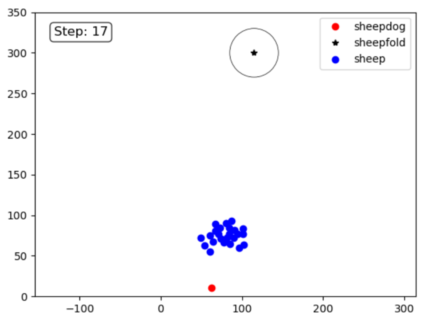

Figure 2. A simple visualization of the starting phase of herding.

Visualization. The more polished and informative visualization component of the project was developed in Unity to render movement data generated from the Python simulation, offering an interactive and insightful format for analyzing the algorithm's behavior. The goal was to create an intuitive environment where the guiding of the herd by the sheepdog under various conditions could be observed.

Path data from the Python simulation was imported into Unity to initiate the visualization. This data, in the form of .json files, represented the movement of each entity (sheep and sheepdog) and enabled the creation of animated paths that reflected their interactions over time. A virtual environment was constructed in Unity to visually simulate the conditions modeled in Python.

The scene was managed by a core object called the "Manager", which dynamically generated all entities based on the .json data. This included determining the number of sheep, dogs, their starting positions, and the destination points. Parameters such as simulation speed and the visualization of entity paths were able to be adjusted during runtime. A graphical representation of the dog and sheep was included, with the option to replace these models.

A flat terrain with textures and painting details was created using Unity's tools to ensure a realistic environment. A skybox featuring a 360° photo of the FRI campus in Ljubljana was included, setting the stage for a more realistic and interesting simulation.

Multiple techniques were employed to highlight critical aspects of the algorithm. Path trails, speed indicators, and field-of-view displays were used to emphasize the dynamics between the sheepdog and the herd. The LineRenderer, assigned by the Manager, rendered each entity's path. As the simulation progressed, entities moved at the same speed from one coordinate to the next, with smooth transitions between points. The rotation of each entity was updated so that they faced the direction in which they moved, using linear interpolation for smooth transitions.

In addition to the core functionality, five different camera modes were implemented, with smooth transitions between them. These modes allow the user to switch between perspectives such as the dog's point of view and a top-down view of the simulation.

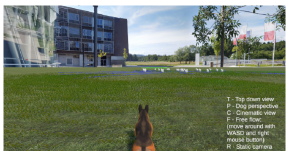

Figure 3. Example of a dog perspective view.

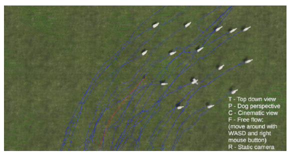

Figure 4. Example of a top down view.

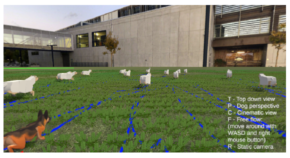

Figure 5. Example of a cinematic view.

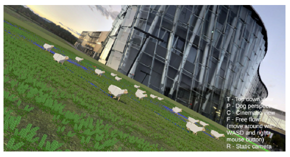

Figure 6. Example of a free flow view.

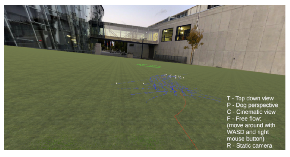

Figure 7. Example of a static camera view.

During the development of the simulation, challenges emerged in ensuring smooth transitions between coordinates and preventing overflows. These challenges were resolved through custom code, which provided full control over the entities' movement and positioning.

Results

The Sheepdog-Driven Algorithm for Sheep Herd Transport was implemented, recreating the core framework described in the original paper. The interactions between the sheepdog and the sheep were modeled using local interaction rules, with the dog designed to guide the herd toward a target destination. Movement dynamics, including attraction, repulsion, and visibility checks, were incorporated, and the agents' positions were updated in real-time to simulate natural behavior.

A 3D visualization was developed to illustrate the algorithm's dynamics, providing an intuitive representation of the interactions between the sheepdog and the herd. The visualization allowed for a detailed analysis of movement patterns and the algorithm's performance under varying conditions.

However, despite closely following the methodology, the simulation did not consistently achieve the intended outcome of guiding the herd to the designated target. While the sheep's movements responded to the sheepdog as expected, the dog was unable to reliably lead the herd to the goal. These results suggest potential areas for refinement in the algorithm's implementation, which are discussed further in the next section.

Discussion

The implementation of the *Sheepdog-Driven Algorithm* revealed several challenges that impacted the consistency of the results. A primary issue was the lack of clarity and detail in the original paper, which introduced numerous terms and variables without sufficient definitions or context.

To give a few examples, the terms $S_r(p_d - x)$ and $S_l(p_d - x)$, introduced in [1], were not explained in terms of their practical computation. Similarly, the variable $D_q^d(k)$, intended to represent the direction from the sheepdog to the sheepfold, was not used in subsequent equations or pseudocode, adding to the ambiguity.

While some terms, such as $o(x) = \frac{x}{|x|}$, were defined mathematically, their interactions with other variables, like $p_q^i(k)$ and $p_d^i(k)$, were not fully clarified. Additionally, the "sheep herd polygon" function $P_s(k)$ was introduced but not explicitly utilized, leaving its purpose unclear.

After following the algorithm as described, unexpected behavior was observed, with the sheepdog initially moving away from the sheep rather than guiding them. A correction was made by reversing the sign in the $o(q(k) - D_l(k))$ function, which resolved the issue and allowed the sheepdog to interact with the herd more effectively. However, no justification for this adjustment was provided in the paper, and it remains unclear if this modification aligns with the intended algorithm design.

The authors were contacted for clarification, but no response was received. This lack of guidance required reliance on trial and error, independent research, and assumptions to address the gaps in the paper. While this approach allowed for partial implementation of the algorithm, some ambiguities remained unresolved. Moreover, the paper itself is not so well-suited for direct implementation, and there is no open-source implementation available.

Despite these challenges, the visualization highlighted the effectiveness of certain behaviors, such as the sheep responding to the sheepdog's position. However, the inability to consistently guide the herd to the target suggests areas for further refinement, including adjustments to the dog's guidance strategy and cohesion dynamics within the herd.

Contributions. IB and SH did the implementation, LP did visualizations, II did the report.

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.