FRIteza

Sheepdog Driven Algorithm for Sheep Herd Transport

Vid Cesar, Gašper Habjan, Matic Hrastelj, and Žan Korošak

Collective behaviour course research seminar report
GitHub: https://github.com/mh4043/CollectiveBehavour-GroupC

December 17, 2023

Iztok Lebar Bajec | associate professor | mentor

At the very early stage of this work, we start to explore the field of collective behaviour. It refers to the study and implementation of algorithms or systems that mimic the self-organized and coordinated actions, observed in natural phenomena. We manage to tackle the concrete problem of sheep herd transport algorithms. First, we present the selected field and make an overview of related works. We then manage to re-create the basic algorithm, which will contribute to new ideas for upgrades, extensions and improvements. In the current phase of our work, we present the methods used so far and the results obtained. We notice that we have already managed to recreate the algorithm to some extent and that we are on the right way.

Collective Behaviour | Swarm Systems | Shepherd | Sheepdog | Herding | Algorithm

The analysis of a sheep flock's behavior serves as an extensively explored example of collective animal behavior. In an effort to reduce individual risk, sheep instinctively move towards the group's center, causing the flock to constrict. There is usually a single agent, which is in most cases a dog, and is tasked with herding a flock of sheep to an enclosure or any desired goal point. However, these sheep act as a swarm system, that tries to evade the shepherding dog, while maintaining flock cohesion [1]. Collective behaviour along with the shepherding problem is very tightly connected with computer and information science even if it might not look like it on the first ball. As a system guidance problem, shepherding has diverse applications, such as robotic sheep herding, crowd control, oil spill cleanups, and safeguarding aircraft. It's also explored in disaster relief, fishing, wildlife management, and handling micro-organisms. In security and military contexts, it's studied for unmanned vehicle maneuvers in combat, terrain searching for intruders, and mine collection [2].

Our work aims to not only replicate the established algorithm by Liu *et al.* [3], but also to provide innovative extensions that enhance its functionality. By leveraging foundational concepts from collective behavior in computer science, our research contributes to the evolution of shepherding algorithms, introducing advancements in efficiency and adaptability.

In the Related work section, we make a brief review of the methods and results on the field of shepherding algorithms. We can notice that this field is very interesting and popular. The authors offer different approaches to useful solutions. In the Methods section, we discuss our concrete methods and algorithms which are explained in detail. In the Results section, we discuss how proposed methods behave in real action, therefore we focus on concrete experimental analysis there. Notice that the following sections may be incomplete due to the early stage of the project.

Related work. Bat-Erdene *et al.* [4] presents an algorithm and design for a multi-robot system engaged in shepherding, demonstrating its applicability in the context of Mongolian herding. It also employs two types of shepherding multi-robots (corner and sideline mobile robots), showcasing their effectiveness in herding sheep. The study introduces a simple system structure and algorithm for controlling these connected mobile robots, emphasizing the need for high mobility to navigate uneven terrain. A very similar problem has been earlier tackled by Miki *et al.* [5], who also ended up proposing a simple but effective shepherding algorithm. By following their method, it is possible to control a flock with about 25 members by a single shepherd and about 30 members by two shepherds. Furthermore, the autonomous cooperation for two shepherds can be generated by the proposed rules.

One of the popular approaches to the challenge we're facing, has also been presented by Strömbom *et al.* [6]. In their work, they suggest a heuristic algorithm that divides the shepherding task into two distinct subtasks: driving and collecting. They observe that, during the process of guiding a herd towards a designated goal, the sheep often exhibit a tendency to separate along the midpoint. In response to this behavior, the authors incorporate an additional subtask dedicated to gathering stray sheep that venture too far from the focal point of the flock.

We have taken work by Liu *et al.* [3] as our starting point. The work mainly deals with the interplay between an agent and a swarm system. A reverse semi-circle recip-

Collective behaviour

In computer and information science, collective behaviour refers to the study of how groups of agents, such as individuals or entities, interact to exhibit coordinated patterns that emerge from their collective actions. This concept finds application in various fields, including artificial intelligence and robotics, where researchers seek to develop algorithms inspired by the collaborative behaviours observed in nature. One concrete application of collective behaviour is the design of sheep herd transport algorithms, which draw inspiration from the coordinated movement of herding animals like sheep. The goal in this specific scenario is to create an algorithm that mimics the way a shepherd guides and controls a flock. For instance, this could enable a swarm system of agents, such as drones or robots, to navigate collectively and efficiently. By understanding and harnessing collective behaviour, researchers aim to develop advanced algorithms that can enhance the coordination, adaptability, and overall performance of swarm systems.

Collective Behaviour | Swarm Systems | Shepherd | Sheepdog | Herding | Algorithm

rocation algorithm is introduced, which is designed to emulate the actions a sheepdog. In proposed algorithm, the sheepdog orients itself towards either the rightmost or leftmost sheep, ensuring the cohesion of the herd while steering it towards the desired destination.

Methods

Our method is designed using the model, originally presented by Liu *et al.* [3].

Firstly we had to choose the proper programming language in which we would later implement and expand the algorithm given in the chosen article. After a brief discussion, we decided that we would use Python, because it is fast, easy to use and offers some basic graphical tools that would suffice for our need (a grid with moving sheep and sheepdog as dots).

Then we had to extract and interpret the necessary equations that the article provides for the algorithm, which also takes into account *vision blocking*. If a sheep is standing in the view distance of the sheepdog and another sheep is standing behind the first sheep inline with the sheepdog, then the sheepdog cannot see the sheep behind the first one. With this in mind, the next paragraph summs up the a few of the basic, but important variables and equations used.

For this algorithm, we need to define the left-hand $\mathbb{P}_l$ and right-hand $\mathbb{P}_r$ side area of a vector x. Then we also need the position $p_i(k)$ and the velocity $v_i(k)$ of the ith sheep and the position $q(k)$ and the velocity $u(k)$ of the sheepdog at kth step. When calculating velocity, rotation parameters and fuzzy logic are used, based on the vision of each entity. We then define the center of the sheep herd $p_c(k)$ and the sheep herd polygon $\mathbb{P}_s$, which defines the boundaries. Furthermore we define the center of the sheepfold p_d and the sheepfold area $\mathbb{P}_d$. The equations are described in detail in the original article and on our GitHub repository. These are all the basics we need to start implementing the algorithm, which uses equations that are based on the described variables and equations.

The algorithm takes as input all the positions (sheepdog, sheep, sheepfold) and other initialization parameters that are used in the calculations of the equations in the original article. The algorithm returns the velocity of the sheepdog. The algorithm runs until all the sheep are in the sheepfold area or until the time runs out. The first part of algorithm checks how many sheep are in the sheepfold area, if there is at least one that it is not in the sheepfold area, then the algorithm continues, otherwise it returns the velocity as 0 and the algorithm finishes. If the algorithm continues, it checks if all the sheep are on the left-hand side of the sheepdog or the right-hand side of the sheepdog. Accordingly to the previus check, it continues to check if the sheepdog is far away enough from the rightmost or leftmost sheep. If it is, it changes the velocity. But if it's close, then it also rotates. At the end, the algorithm returns the updated velocity of the sheepdog.

Improvements. In the base paper, we have found several different ways to improve sheepdog driven algorithm or to robustify it even more for different specific situations. These improvements can be classified as performance improvements that improve specific results and fix specific issues, or improvements that make the algorithm more similar to a real life sheepdog performing its job. One specific example of a performance improvement is an edge case, where the sheepdog will not detect any sheep in a certain step, due to its detection radius being smaller than the distance to the closest sheep. The algorithm does not quite have an answer to the question "What happens when sheepdog is too far away from sheep?". This will in turn lead the algorithm to a premature stop, since the sheepdog will not be able to choose any sheep to chase. To solve this issue, we have designed a "fail-safe" workaround, which will send the dog into a search for sheep, whenever this occurs. We are approaching this problem in a few different ways, which determine what route the dog is going to take to start the search. One of the ways we are looking in to include this imporvement is by studying and perhaps later implementing the wandering algorithm introduced by Reynolds in [7].

Results

In accordance with the described methods, for the first step we tried to recreate the algorithm. In the implementation itself, we have omitted some details for now. We used programming language Python, as it is very comfortable for writing such algorithms. Let us present the obtained results.

Regarding the equations in the article, we found out that some of them are not fully described, so it was a bit harder to understand, and at least two equations include a part that doesn't really contribute to the outcome of the equation. For ex-

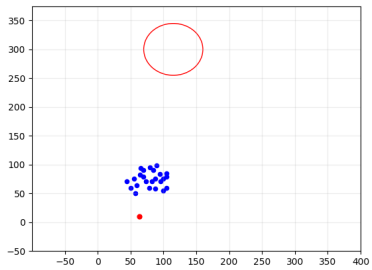

Figure 1. Initial positions of the agents (step 0).

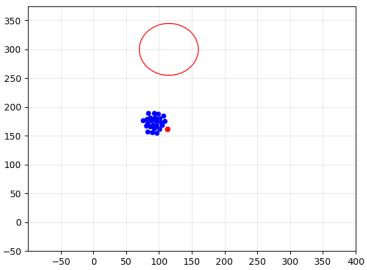

Figure 2. Step 666 of the algorithm.

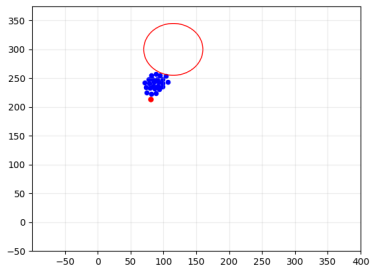

Figure 3. Step 1116 of the algorithm.

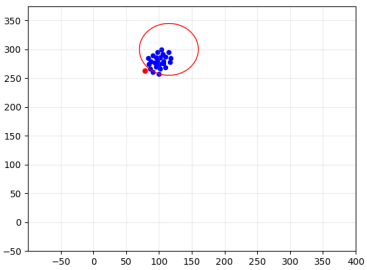

Figure 4. End of the algorithm (step 1584).

ample the equation (25) in the article uses the length of a unit vector, which is 1, to calculate the product.

Figure 1 represents the initial positions of the sheepdog (red dot), sheep (blue dots) and the radius of the sheepfold (red circle), where the sheepdog has to herd all of the sheep. Figures 2 and 3 represent the 666. step and the 1116. step respectively. And figure 4 represents the final step of the algorithm, where all of the sheep are in the sheepfold (based on their center).

Discussion

So far we have managed to almost fully re-create the basic algorithm, proposed in the baseline article, by Liu *et al.* [3], but we have omitted some details at implementation for now. For now, we are satisfied with the results obtained. Sheepdog successfully accomplishes his job, which is to lead sheep to the declared goal point. However, there is a variety of options to improve behaviour, but the algorithm will serve as a good springboard for further work.

We have also already come up with some further suggestions for improvements and expansions. For instance, the concept of attracting rather than repelling a subset of sheep presents a straightforward adjustment in the equation, offering an interesting avenue for practical implementation. Additionally, the introduction of obstacles within the field also emerges as a realistic and valuable extension, providing insights into the algorithm's adaptability in diverse environments. The envisioned modifications could be later on expressed through mathematical equations. These potential upgrades not only enrich the fundamental algorithm, but also pave the way for practical experimentation to observe their effectiveness and applicability.

CONTRIBUTIONS. The analysis of the baseline paper was done by all members together. GH wrote the introduction, made a review of related works and in the end discussed current progress. MH described the methods, facilitated the interpretation of the equations and done some research on the wandering algorithm by Reynolds. ŽK and VC took care of the actual implementation of the algorithm and summarized the results. Polishing of the report based on comments was done by all members together.

Bibliography

1. Lien JM, Bayazit O, Sowell R, Rodriguez S, Amato N (2004) Shepherding behaviors in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004.* Vol. 4, pp. 4159–4164 Vol.4.
2. Long NK, Sammut K, Sgarioto D, Garratt M, Abbass HA (2020) A comprehensive review of shepherding as a bio-inspired swarm-robotics guidance approach. *IEEE Transactions on Emerging Topics in Computational Intelligence* 4(4):523–537.
3. Liu Y et al. (2021) Sheepdog driven algorithm for sheep herd transport in *2021 40th Chinese Control Conference (CCC).* pp. 5390–5395.
4. Bat-Erdene B, Mandakh OE (2017) Shepherding algorithm of multi-mobile robot system in *2017 First IEEE International Conference on Robotic Computing (IRC).* (IEEE), pp. 358–361.
5. Miki T, Nakamura T (2006) An effective simple shepherding algorithm suitable for implementation to a multi-mmobile robot system in *First International Conference on Innovative Comput-*

ing, Information and Control - Volume I (ICICIC'06). Vol. 3, pp. 161–165.
6. Strömbom D et al. (2014) Solving the shepherding problem: heuristics for herding autonomous, interacting agents. *Journal of the royal society interface* 11(100):20140719.
7. Reynolds CW, et al. (1999) Steering behaviors for autonomous characters in *Game developers conference*. (Citeseer), Vol. 1999, pp. 763–782.