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

January 7, 2024

Iztok Lebar Bajec | associate professor | mentor

We explore the field of collective behaviour in this work. 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. Firstly, we successfully re-create one of the existing algorithms in the field of shepherding. We then present the methods used and the results obtained. Afterwards, we develop some new ideas for upgrades, extensions and improvements. In the end, we create few variations of the basic algorithm, by applying some minor improvements. In each variation of the algorithm, sheepdog successfully leads sheep to the desired goal.

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 (sheepdog), 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. 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.

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 reciprocation algorithm is introduced, which is designed to emulate the actions of 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.

We can notice that this field is very interesting and popular. The authors offer different approaches to useful solutions.

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

Methods

As mentioned previously, we've chosen and implemented the model presented by Liu *et al.* in [3]. In the following subsections we take a look at how the original model works and what are its' core parameters and equations. Then, we go through the process of how we implemented presented model (workflow). At the end, we provide explanations on how we implemented the improvements, which we decided for – combing and two sheepdogs.

Model. Firstly, we need to define the motion of both entities. The equations $p_i(k+1) = p_i(k) + Tv_i(k)$ and $q(k+1) = q(k) + Tu(k)$ denote the motion of the i-th sheep and sheepdog respectively, where $p_i(k)$ and $q(k)$ denote the position, $v_i(k)$ and $u(k)$ denote the velocity, and T denotes the sampling period. From this statement, we get the displacement between the i-th sheep and the sheepdog with the following equation: $p_i^q(k) = p_i(k) - q(k)$. We can divide the equation for velocity of the i-th sheep $v_i = v_{di}(k) + R(\theta_i(k))v_{si}(k)$ into three parts. The $\theta_i(k)$ is the rotation angle, R is the rotation matrix, $v_{di}(k)$ represents the reaction of the sheepdog and $v_{si}(k)$ represents the reaction of other sheep. Detailed equations of these three parts are available on our GitHub repository and in the original article.

The model also takes into account vision blocking between sheep and the sheepdog. If the i-th and j-th sheep have the same heading, and if the distance between the j-th sheep and the sheepdog is smaller than the distance between the i-th sheep and the sheepdog, then the i-th sheep is *vision-blocked* by the j-th sheep from the viewpoint of the sheepdog. If the distance between the j-th sheep and the sheepdog is smaller than the vision radius of the sheepdog (ρ_v), then that sheep is *visible* to the sheepdog.

Following that, we can calculate the estimated center of the sheep herd $p_c(k)$, by averaging the positions of visible sheep with the number of visible sheep. We then construct the sheep herd polygon $\mathbb{P}_s(k)$ and the sheepfold $\mathbb{P}_d$ with the following equation: $\mathbb{P}_d = \{p | p \in \mathbb{R}^2, ||p - p_d|| \leq \rho_o\}$ (a set of positions that are in the sheepfold), where ρ_o is the radius of the sheepfold and p_d is the center of the sheepfold.

Finally, the task is to calculate the $u(k)$ (velocity of the sheepdog), such that for the initial condition of $\mathbf{d}(q(0), \mathbb{P}_s(k)) > 0$ (distance between sheepdog and sheep herd polygon), we get $\mathbf{d}(p_i(k), \mathbb{P}_d) = 0$ (distance between sheep and sheepfold), for every sheep at the end of the algorithm.

These are all the basics. The more complex equations of parameters are available and described on our GitHub page and in the original article.

Implementation of the algorithm. We decided to use programming language Python, because it is somewhat fast, easy to use and offers some basic graphical tools that would suffice for our need.

The algorithm takes as input a plethora of user defined and initialization parameters. Most of them are used as weights or thresholds in equations. It also takes as input all of the positions of entities (sheep, sheepdog, sheepfold) and their radii. Inside the loop, at each step, the algorithm checks if the goal has been reached, i.e. all the sheep are in the sheepfold area. If there is at least one that hasn't reached the goal, the algorithm calculates the velocity of the sheepdog(s), where it checks whether all sheep have reached their goal. If yes, then the sheepdog stops, else the algorithm calculates all the visible sheep and the center point of them. If the sheepdog has no sheep in sight, then the combing mechanism kicks in. Combing is described in the next subsection. But if the sheepdog sees at least one sheep, then the algorithm checks if all the visible sheep are on the right side of the sheepdog, if so, it then gets the right most visible sheep and by using one of the threshold parameters, it decides whether to chase or not. And vice versa if the sheep are on the left hand side. After calculation of velocity, the sheepdog then moves in the calculated direction. The algorithm then calculates the velocities of the sheep. If a sheep has reached its' goal, then it just slows down, else the velocity gets calculated using the equations, described in the previous subsection. After calculation of velocity, the sheep then move in the calculated direction. The algorithm stops if all the sheep have reached their goal or if the number of steps have surpassed the maximum amount of steps.

Improvements. For the improvements, we decided to implement the combing mechanism and two sheepdogs, and then compare the results with the basic algorithm.

Combing. In the basic algorithm, if we have 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, it will 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 sheepdog into a search for sheep.

The sheepdog will start going downwards, until it sees the edge. It will then turn 90° to the to the left and go in that direction for the length of its vision radius. Then, it will stop and turn 90° to the left again, and will go upwards until it sees an edge. There, it will turn 90° to the right and go in that direction for the length of its vision radius. After that, it will turn 90° to the right and redo the cycle. When it reaches the edge, it will turn back and start going the other way. If at any point sees a sheep, it will stop combing and start chasing it.

Two sheepdogs. The other improvement is the ability to use two sheepdogs. Here, we basically just added another sheepdog in a different position on the sheep fold. There is the same logic behind them when it comes to calculating velocity and moving.

Results

Figure 1 represents initial positions of the sheepdog (red dot), radius of the sheepdog (cyan circle), 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 step 3580 and step 4460 respectively. Figure 4 represents the final step of the algorithm, where all of the sheep are in the sheepfold (based on their center). From the first two figures, we can see combing in action, where the sheepdog firstly combs the area until it sees the first sheep.

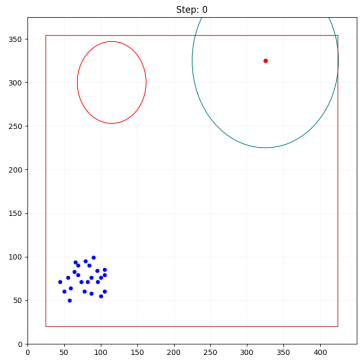

Figure 1. Initial positions of the agents (step 0).

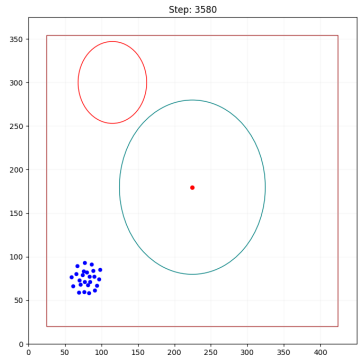

Figure 2. Step 3580 of the algorithm.

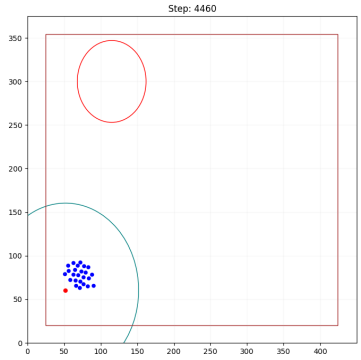

Figure 3. Step 4460 of the algorithm.

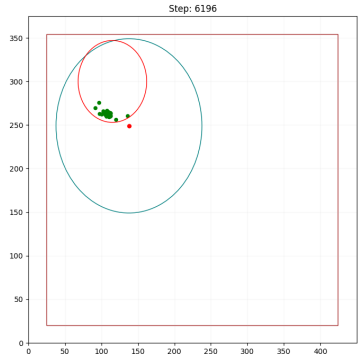

Figure 4. End of the algorithm (step 6196).

Figures 5, 6, 7 and 8 represent the extension to the basic setup, that is incorporation of two sheepdogs (purple and red dot). The number of steps is usually expected to be lower with two sheepdogs, but that is not always the case. When they start coming closer to each other, the number of steps is similar to the basic algorithm with one sheepdog. The main advantage takes place when we have sheep spread around the map and sheepdogs start further apart.