Aalto University
School of Electrical
Engineering

Master’s programme in ICT Innovation

Realistic Physical Simulation and Analysis of Shepherding Algorithms using Unmanned Aerial Vehicles

Kishore Kumar

Master’s Thesis
2023

Aalto University
School of Electrical Engineering

Author Kishore Kumar

Title Realistic Physical Simulation and Analysis of Shepherding Algorithms using Unmanned Aerial Vehicles

Degree programme ICT Innovation

Major Autonomous Systems

Supervisor Prof. Quan Zhou

Advisor Mr Marco Moletta (MSc)

Collaborative partner Flox AB

Date 17 July 2023 **Number of pages** 62 **Language** English

Abstract

Advancements in UAV technology have offered promising avenues for wildlife management, specifically in the herding of wild animals. However, existing algorithms frequently simulate two-dimensional scenarios with the unrealistic assumption of continuous knowledge of animal positions or involve the use of a scouting UAV in addition to the herding UAV to localize the position of the animals. Addressing this shortcoming, our research introduces a novel herding strategy using a single UAV, integrating a computer vision algorithm in a three-dimensional simulation through the Gazebo platform with Robot Operating System 2 (ROS2) middleware. The UAV, simulated with a PX4 flight controller, detects animals using ArUco markers and uses their real-time positions to update their last known positions.

The performance of our computer-vision-assisted herding algorithm was evaluated in comparison with conventional, position-aware/dual UAV herding strategies. Findings suggest that one of our vision-based strategies exhibits comparable performance to the baseline for smaller populations and loosely packed scenarios, albeit with sporadic herding failures and performance decrement in very tightly packed flocking scenarios and very sparsely distributed flocking scenarios. The proposed algorithm demonstrates potential for future real-world applications, marking a significant stride towards realistic, autonomous wildlife management using UAVs in three-dimensional spaces.

Keywords UAVs, PX4, Simulation, Computer Vision , Aruco Markers, Furthest agent targetting controller, Collect and drive algorithm, Herding, Gazebo, ROS2

Preface

The successful completion of this thesis owes much to the valuable guidance, support, and commitment of several individuals, to whom I express my sincere gratitude. Foremost, I extend my profound appreciation to my thesis advisor, Marco Moletta, Robotics Lead at Flox AB, and my thesis supervisor, Prof. Quan Zhou from Aalto University, whose expert insights and direction have been instrumental throughout this academic journey.

I would also like to convey my heartfelt thanks to the entire team at Flox AB, specifically CEO Sara Nozkova and Matteo Tadiello, the AI and Autonomous Lead. Their sustained support, from providing necessary resources to creating conducive conditions, played a pivotal role in meeting the requirements of this thesis. My gratitude further extends to EIT Digital for the vital introduction to Flox, a critical step marking the commencement of my professional career.

Their collective contributions and encouragement have enriched my thesis work, broadened my academic horizons, and for that, I am immensely grateful.

I also want to thank my friends and family for providing emotional support during the hardships faced during this time period.

Otaniemi, 17 July 2023

Kishore Kumar

Contents

Symbols and abbreviations

Symbols

m_{c}	cohesion multiplier co-efficient 1
m_{cp}	cohesion multiplier co-efficient 2
m_{a}	alignment multiplier co-efficient 1
m_{ap}	alignment multiplier co-efficient 2
m_{s}	separation multiplier co-efficient 1
m_{sp}	separation multiplier co-efficient 2

Abbreviations

CD	Collect and Drive
COM	Center of Mass
FAT	Furthest Agent Targetting
GCM	Global Center of Mass
ROS	Robot Operating System
RTF	Real-Time Factor
SDF	Simulation Description Format
SITL	Software-In-The-Loop
SONAR	Sound Navigation and Ranging
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle

1 Introduction

This thesis aims to develop a UAV-based herding algorithm that leverages computer vision to deter wild animals from agricultural lands. The goal is to provide a cost-effective, efficient, and environmentally-friendly solution to the significant issue of wildlife-induced damage in agriculture.

1.1 The Herding Problem

The issue of wildlife interference in agricultural activities has been a longstanding problem, posing significant challenges to farmers worldwide. The severity of this problem is particularly evident in the field of agriculture, where wildlife, notably wild boars, moose, red deer, and fallow deer, geese, and whooper swans, have been causing substantial damage to crops. According to a report by Statistics Sweden, wildlife destroyed a staggering 165,000 tonnes of grain in 2020 [1]. This figure is nearly double the 88,000 tonnes reported in 2014 [1], highlighting the escalating nature of this problem.

The impact of wildlife on agriculture extends beyond mere crop destruction. It also influences the choice of crops grown by farmers. More than one in three farmers stated that the presence of wildlife affects their choice of crop [1]. This suggests that the wildlife problem is not only causing direct damage to crops but also indirectly influencing agricultural practices and decisions.

Given these alarming statistics, it is evident that wildlife encroachment is a pressing issue that requires immediate attention and effective solutions. The task of herding or repelling wildlife away from agricultural fields, known as the "herding problem," is a complex challenge that involves various factors, including the behavior of wildlife, their interaction with a predator, and more. Addressing this problem is crucial to safeguard agricultural productivity, protect wildlife, and ensure the sustainability of farming practices.

1.2 Evolution of Strategies in Tackling Herding

Using unmanned aerial vehicles (UAVs) or unmanned ground vehicles (UGVs) to herd animals proves to be a more benign approach compared to hunting invasive species, as it circumvents any harm inflicted on wildlife. Moreover, there exists the potential for complete automation, thereby eradicating the need for manual labor. Over the years, many solutions have been proposed to tackle the herding problem. The trajectory of herding methodologies has witnessed significant advancements, commencing with Vaughan et al's seminal work in 1998 [12]. This innovative study employed a terrestrial robot to shepherd ducks, utilizing potential fields where virtual forces dictated the robot's movement.

Subsequent research predominantly utilized 2D simulations, with points symbolizing the predator and animal agents. A notable deviation was Paranjape et al's work, which actually deployed the algorithm in the real world by employing UAVs to deter birds from airspace using a mathematical model of avian behavior [13].

Strömbom et al introduced a 'collect and drive' algorithm, guiding the predator to aggregate animals and direct them towards a goal [6]. Despite real-world attempts, this strategy did not incorporate Unmanned Ground Vehicles (UGVs) or UAVs [11].

Gade et al proposed a graph-based strategy to regulate flock size and guide them to a goal [7]. These methodologies predominantly involved a single herding agent, presupposing knowledge of all animal positions, an unrealistic assumption in practical scenarios. Li et al proposed a swarm of predator UAVs for herding [9], but this too required prior knowledge of animal positions.

Addressing this, King et al suggested a second 'Surveillance' UAV to monitor animal and predator positions [10]. This necessitated two UAVs for herding, increasing costs.

These promising strategies, however, are economically unfeasible due to the requirement of multiple UAVs. Hence, a single UAV-based solution utilizing a camera for animal localization is needed.

The most viable solution thus far was proposed by Tsunoda et al [22], where a terrestrial robot herds animals within its field of view. However, this strategy does not track past-seen animals and is limited by its terrestrial perspective, underscoring the need for an aerial UAV-based solution.

1.3 The Research Question

In light of the limitations identified in previous herding strategies, there is an urgent need for a solution that is not only more feasible for real-world deployment but also more effective in managing the herding problem. A promising approach to address these challenges is the implementation of a single-UAV herding strategy, which capitalizes on an aerial perspective and an enhanced field of view. This strategy is also applicable to situations in which the animals may move out of the UAV's field of view.

The proposed solution in this thesis employs a single UAV equipped with a camera and gimbal to perform herding without the need for a surveillance UAV. It utilizes the "Collect and Drive" algorithm proposed by Strömbom et al [6] and the "Furthest Agent Targeting" controller introduced by Tsunoda et al [22]. Initially, herding is performed under the assumption that the position of all animals is known beforehand, as per the original 2D papers, and the performance of these algorithms serves as the baseline for comparison.

The primary research question and the sub-questions that this thesis seeks to answer are as follows:

- **"How does the performance of the proposed single-UAV herding strategy compare to the baseline performance of the existing two-UAV herding strategies?"**
  - "How does the success rate of the proposed single-UAV herding strategy compare to the baseline strategies?"
  - "How does the time efficiency of the proposed single-UAV herding strategy compare to the baseline strategies?"

  - "How does the travel distance of the UAV when using the proposed single-UAV herding strategy compare to the baseline strategies?"
  - "What is the herding capacity of the proposed single-UAV herding strategy?"

The answers to these questions will provide a thorough evaluation of the proposed single-UAV herding strategy, offering valuable insights into its time efficiency, success rate, herding capacity, and travel distance. This comprehensive analysis will contribute to our understanding of the strategy's potential for real-world application and its effectiveness in addressing the herding problem.

1.4 Scope and Constraints

The boundaries of this thesis are defined by specific parameters and are subject to certain limitations, which are essential to acknowledge for a comprehensive understanding of the research.

One major limitation is the computer vision algorithm used to determine the animals' positions on the ground. This thesis does not go into the details of this algorithm. Instead, it adopts a simpler approach that combines an object detection algorithm with the actual animal positions. This method serves as a substitute for computer vision-based triangulation, potentially paving the way for future research to explore its implementation further.

Another significant limitation is the testing environment for the proposed algorithm. Due to constraints related to development costs, time, and the inherent risks associated with deploying this algorithm in the real world without comprehensive safety measures, the algorithm has been tested exclusively in a simulation environment. While the results derived from these tests are promising, they have not been validated in a real-world setting. This limitation underscores the need for future research to bridge this gap, thereby enhancing the practical applicability and robustness of the proposed single-UAV herding strategy.

These constraints do not undermine the value of the research but rather delineate its scope, providing a clear direction for future work to build upon the foundation laid by this thesis.

1.5 Thesis Structure

This thesis is organized into distinct sections, each serving a specific purpose and collectively contributing to the overarching narrative of the research.

- The **Introduction** sets the stage for the study, outlining the background, identifying the problem statement, articulating the research question, and delineating the scope and constraints of the research. It provides the necessary context and frames the research objectives.

- The **Literature Review** offers a comprehensive overview of the existing body of knowledge on the topic. It highlights key findings from previous studies, identifies gaps in the current understanding, and underscores the need for the present research.
- The **Research Material and Methods** section provides a detailed account of the methodology employed in the study. It describes the simulation environment, elaborates on the algorithms used, and outlines the testing procedures, ensuring transparency and reproducibility of the research.
- The **Results** section presents the empirical findings of the study. It provides a comparative analysis of the performance of the proposed single-UAV herding strategy and the baseline two-UAV strategies, offering quantitative evidence to address the research question.
- Finally, the **Discussion** section offers a thoughtful interpretation of the results. It contextualizes the findings within the broader research landscape, discusses the implications of the results, and suggests potential avenues for future research.

2 Related Work

2.1 Robotics Herding

The herding problem, as described by Lien et al. [3], is a specific type of flocking behavior where a guiding agent (or agents) is used to control the movement and direction of a group of flock agents. The guiding agent is represented by a robot, and it exerts an external force that influences the flock agents, causing them to move away from the guiding agent. In real-world scenarios, this influence can be achieved through various means such as loud noises, blinding lights, or frightening maneuvers to scare the animals away. The guiding agent employs strategies and movements to steer the flock agents from their starting position to a goal position, including the use of planned routes and tactics to regroup separated flock agents. The guiding agent can be any autonomous robot capable of navigating aerial, aquatic, or terrestrial environments. In this thesis study, the guiding agent is the UAV, and the flock agents are the representations of animals that need to be guided to the goal position using the UAV.

The domain of robotic animal herding brings together biological sciences, robotics, and computer science. The ability to direct and regulate the movement of animal groups through autonomous systems has significant implications for sectors such as agriculture, wildlife management, and environmental preservation. Over the years, research in this field has evolved, with each study building upon the insights of its predecessors.

Initial explorations in this field involved the use of a terrestrial robot to shepherd small assemblies of ducks within a confined arena [12]. This seminal study employed simulation to evaluate the effectiveness of the herding algorithm before its application to actual ducks, highlighting the important role of simulation testing in this domain. The results obtained from the real-world experiment aligned with those from the simulation, confirming the value of simulations in this context.

Simulations are particularly advantageous when UAVs are deployed for herding purposes, given the significant costs and potential risks associated with these devices. Therefore, it is prudent to first validate the herding algorithm within a secure and controlled simulation environment before its real-world deployment.

However, before delving into the specifics of the simulations used in the aforementioned experiment, it is crucial to understand the fundamental components required for a herding simulation. This literature review will sequentially navigate through the contributions that have facilitated the development of a vision-based UAV herding simulation. A comprehensive summary of the research trajectory in the field of herding is provided by Long et al. [8].

2.2 Simulating Flock Behaviour

The pivotal work by Craig W. Reynolds [2] has been instrumental in the field of simulating flocking behavior. Reynolds introduced the term 'boids' to refer to bird-oid or bird-like objects. These boids adhere to simple movement behaviors based on their local perceived environment, leading to the emergence of complex group behaviors. The model is based on three fundamental rules: collision avoidance, velocity matching, and flock centering.

Collision avoidance ensures that each boid maneuvers to maintain a minimum distance from other boids in its vicinity. Velocity matching involves each boid aligning its direction and speed with those of the boids in its vicinity. Flock centering drives each boid towards the perceived center of mass of the boids in its vicinity.

Each 'boid' is implemented as an autonomous actor that navigates based on its local perception of the dynamic environment. The significance of this paper lies in its introduction of the 'boids' concept and the provision of a simple yet effective model for simulating flocking behavior. This model has found extensive application in computer graphics and animation and has also been utilized in the study of animal behavior.

While the three parameters of Reynolds' model effectively simulate flock behavior, they cannot be directly applied to herding simulations. This is primarily due to the need to simulate animal behavior in the presence of a predatory entity. Thus, the Reynolds Flocking model by itself cannot be used for the simulation of a herding environment containing animals and a predator.

In real-world scenarios, animals use pheromones to communicate emotions throughout the flock [4]. This concept was integrated into simulation by Barksten et al., who introduced the 'Escape rule' [5]. In the foundational flocking behavior model, the rules for cohesion, separation, and alignment generate vector outputs. These output vectors are multiplied by a set of scalar coefficients and subsequently summed together to produce the resultant motion vectors of the animals. The Escape rule introduces an additional multiplier that amplifies the original three rules in the presence of a predator. Further modifications include the introduction of a human-controlled predator, the adaptation of the modeled animals to move in two dimensions instead of three, and the alteration of their behavior to mimic sheep rather than birds.

A simulator was developed using the modified version of Reynolds' flocking model to compare the behavior of simulated sheep with real sheep, primarily using quantitative data from the simulator. The comparison revealed several similarities between the simulated and real sheep, suggesting that the extended Reynolds model can serve as a foundation for this scenario.

The amalgamation of the behavior of animals reacting to others within their flock and their behavior in the presence of a predator establishes the basic environment required to perform herding using a UAV. However, this experiment was carried out in a simulation where the herding agent had access to the positions of all the animals present in the simulation, which makes it unrealistic and challenging to deploy in the real world.

2.3 Simulating Animal Herding

The groundbreaking Robot Sheepdog project by Vaughan et al. [12] employed a ground-based robot to shepherd a flock of ducks. The methodology involved creating a minimal simulation model of the ducks' flocking behavior, which subsequently informed the design of a flock-control algorithm. This algorithm was based on the

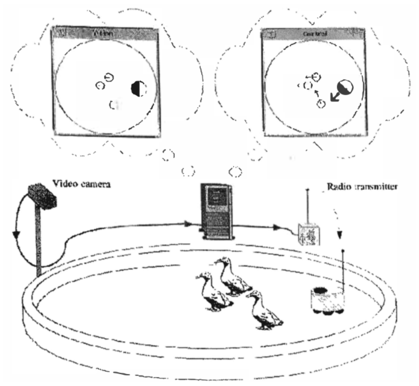


Figure 1: Robot Sheepdog system overview.

concept of forces acting on particles, with the robot being attracted to the center of the flock and repelled from both the flock center and the goal position.

This approach was initially tested in a simulation before being implemented on a physical robot within a real-world environment. The robot's position, orientation, and movement were determined by processing a video image stream, with real-time adjustments made via a radio modem, as shown in Figure 1. This experiment proved that it was indeed possible to conduct experiments in simulation and deploy them in the real world with minimal changes, provided the simulation was modeled closer to reality.

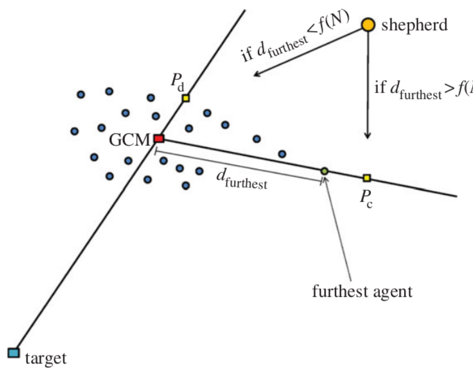
The work conducted by Kachroo et al. [17] is significant within the domain of dynamic programming solutions for herding problems. Their research focused on developing three dynamic programming algorithms: Admissible Policy Search Technique, Value Iteration Technique, and Policy Iteration Technique. These algorithms aim to guide a single "evader" agent, using a pursuing agent, toward a target location. In this study, the herding problem is formulated as a cost optimization problem, with the objective of herding the evader agent in the shortest possible time. The study discovered that one algorithm was time-consuming, while the other two strategies converged to the same optimal cost value functions. However, a major drawback of this approach is its limited discussion on using these techniques solely for solving the cost value function while searching for the optimum control policy in a single pursuer-evader problem. Parallel to this, the study conducted by Miki and Nakamura [18] presented a straightforward yet effective algorithm for simulating a herding task. The behavior of the swarm was governed by a set of four rules, which included cohesion, separation, and escape rules similar to the study conducted by Barksten et al. [5] to simulate prey behavior. However, instead of using the alignment rule, the fourth rule involved a random action that induced stochastic behavior. An individual within a group responds solely to the other individuals in its immediate vicinity.

The practice of shepherding relies on a set of rules: Guidance, Flock Making, Keeping, and Cooperation. These rules dictate how shepherds guide a flock, bring stray individuals back to the group, maintain distance from flock members to preserve order, and coordinate with other shepherds to avoid overlap. Shepherds react to other shepherds within a limited area to prevent overlap and search for flock members within their nearby surroundings. This paper introduces the terms "handling zone" and "watching zone" to refer to these respective areas. The shepherd's movement is determined by the angle (θ) between the desired direction and the object's current direction. However, controlling UAVs using this approach is not ideal as it relies on controlling only the yaw angle of the drone, while having a constant velocity. UAVs can easily maneuver in multiple directions. Additionally, the use of position sensors such as ultrasonic sensors and IR sensors can be susceptible to false positives, causing the predator to mistake other objects for sheep.

The research conducted by Strömbom et al. [6] devised a simple heuristic for replicating sheep behaviors, employing a single shepherd. Their paper introduces the Collect and Drive (CD) algorithm for herding. The algorithm, grounded in simple heuristics, illustrates how a shepherd can guide autonomous, interacting agents towards a designated target. Figure 2 illustrates the algorithm.

The shepherd adaptively switches between collecting agents when they are excessively dispersed and driving them once they have been aggregated. This strategy minimizes the likelihood of the group splitting, enabling the shepherd to maintain the group's movement towards the target location. However, the CD algorithm is not guaranteed to succeed when agents interact with less than half of the total group size. Under such circumstances, the group of agents is prone to splitting into two or more stable subgroups.

To circumvent this issue, the shepherd is programmed to sequentially herd subgroups of a size it can manage. This is achieved by applying the same shepherding algorithm to the local center of mass of nearby agents, as opposed to the global center of mass. This nuanced approach allows for more effective control of the group, even under challenging conditions. One drawback observed in this approach is that while the shepherd switches from the collect phase to the drive phase, it often moves through the flock instead of moving around it, thereby breaking the flock again. This inefficiency results in the shepherd wasting a lot of time trying to gather the agents together.



if $d_{\text{furthest}} < f(N)$
shepherd
P_d
if $d_{\text{furthest}} > f(N)$
GCM
d_{furthest}
P_c
furthest agent
target

Figure 2: Overview of the collect and drive algorithm: The Shepherd switches between the collect and drive positions if the radius of the flock with center GCM exceeds a particular value, $f(N)$, which is a function of N, the number of sheep present.

Expanding on Strömbom et al.'s work [6], Hoshi et al. [19] explored the effects of altering the step size and introducing three-dimensional motion in herding tasks [20]. Concurrently, Lee and Kim [21] developed an alternative set of behaviors for the shepherd and swarm, demonstrating the adaptability of herding algorithms. Tsunoda, Sueoka, and Osuka [23] added an angle of error to the shepherd-swarm interaction, introducing unpredictability into the dynamics.

The study by Fujioka and Hayashi [24] introduces a fresh perspective to the shepherding problem. The problem, which involves guiding a herd of sheep from their initial position to a target position, has wide-ranging applications, including robotics and crowd control. The authors propose new shepherding behaviors, namely V-formation control and GCM+V control, and compare these with a previously proposed method, GCM-targeting.

Gade et al. [7] present the n-wavefront algorithm, a boundary control strategy that enables a single Unmanned Aerial Vehicle (UAV) to herd a flock without causing fragmentation. The algorithm selects "n" boids on the flock's boundary to be influenced by the UAV, thereby preventing panic within the flock. The selection of these boundary boids is based on user-defined priorities, such as boundary keeping (influencing boids closest to the Protection Zone), herding (influencing boids farthest from the desired location), or a hybrid approach. The authors introduce a parameter λ to represent the ratio of importance given to boundary keeping and herding, suggesting that other metrics, such as centrality, could be used to select influential nodes. The effectiveness of the n-wavefront algorithm is demonstrated through numerical simulations. However, similar to other papers, this algorithm also requires prior knowledge of the positions of the animals.

Paranjape et al. [13] advanced the n-wavefront algorithm by proposing the m-wavefront algorithm. This herding strategy was developed to address the challenge of herding a flock of birds. The algorithm involves the selection of m waypoints, which are points in the environment that the UAV aims to reach. The UAV moves towards these waypoints in a sequence, thereby influencing the movement of the flock. The selection of these waypoints is based on the current state of the flock and the desired goal state. The algorithm assumes that the pursuer has access to real-time data about the flock, such as that provided by avian radars. While this makes it possible to use a single drone to herd the animals, this concept cannot be translated to herd terrestrial and aquatic animals. This algorithm also focuses on deflecting a flock of birds from its current direction to avoid flying over a restricted airspace rather than herding the birds to a desired location. This approach is not useful for guiding animals from a farmland to a location in a nearby forest, which is what our algorithm focuses on.

Li et al. [9] proposed a novel approach to herding animals using drones that emit barking sounds. The drones are designed to move in a way that mimics the behavior of a herding dog. The spread range of the barking from the drone is fan-shaped, and only animals within this range will be affected by the repulsion of the barking drone. The paper introduces two guidance laws, namely Fly-to-edge and Fly-on-edge, for navigating a barking drone from one point to another in the shortest time. The paper also presents the optimal positions (steering points) for the barking drones to efficiently gather animals, as well as the collision-free allocation of the steering points. The proposed method is shown to be quite robust and not sensitive to parameter variations

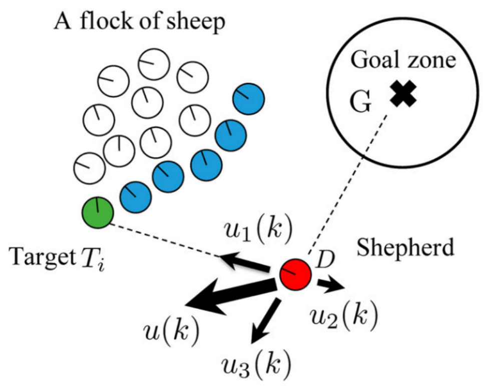


Figure 3: Overview of the furthest agent targeting algorithm: The Shepherd experiences forces of attraction (u_1) and repulsion (u_2) from the target animal T_i to approach it while maintaining a safe distance from it. It also experiences a repulsive force (u_3) from the Goal position G. The net force results in the shepherd approaching the target T_i from its behind to push it towards the goal zone.

and uncertainties of the control channel. This algorithm introduces the concept of using directional cones within which the animals are scared, which is similar to what we would expect in the real world, as animals are more afraid of sounds emitted from a speaker attached to a UAV as opposed to simply running away due to the mere presence of a predator UAV. However, this approach relies heavily on the use of multiple drones, which is not desirable for our application.

Tsunoda et al. [22] introduced a unique approach to shepherding navigation that relies on a local camera-based system. This method stands out from most of the previous techniques as it doesn't necessitate the knowledge of the positions of all animals in the herd. Instead, it requires the shepherd to perceive the positions of the animals within its field of view. This is illustrated in Figure 3.

The authors propose a shepherd-like navigation system where the shepherd attempts to herd a group of animals from the side. The shepherd targets the farthest animal in its field of view and attempts to move it towards the desired direction, an approach inspired by the real movement of a sheepdog when herding real sheep.

The paper presents statistical analysis results, demonstrating that the farthest-agent targeting control is effective if the animal group is loosely gathered. It also shows that shepherd navigation succeeds even if the shepherd cannot perceive the positions of all animals. The authors further explore navigation scenarios where the target animal is selected from the farthest group and where positional errors for animal positions are included. They conclude that navigation based on the proposed method succeeds even though the shepherd is placed across the goal from the flock or is placed inside the flock. This algorithm shows the most promise and is suitable for our research. Thus, our proposed algorithm seeks to augment this algorithm for the case of aerial herding UAVs.

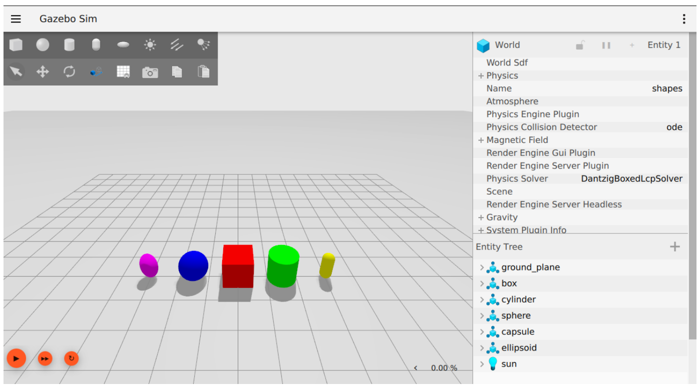
3 Research Material and Methods

This section provides an overview of the methods used and outlines the practical steps taken in our research to address the challenges identified in the domain of wildlife herding and repellence. We detail our simulation environment, UAV and camera specifications, and the characteristics of our simulated animals. We also explain our position update algorithm, the conditions for switching from patrolling to herding, and the baseline algorithms used for comparison. Finally, we present our proposed herding algorithm and the conditions for the termination of the herding process. This methodology provides the details necessary to ensure the reproducibility of this framework for future research in this field.

3.1 The Simulation Environment Setup

3.1.1 The Gazebo simulator and the PX4 autopilot

Gazebo is a powerful open-source 3D robotics simulator that provides a realistic environment for testing and development. It integrates robust physics engines, high-quality graphics, and convenient programmatic and graphical interfaces, making it a valuable tool for simulating robots. It allows developers to rapidly test algorithms, design robots, perform regression testing, and train AI systems using realistic scenarios. Gazebo does provide a variety of pre-built environments, or "worlds", that can be used for simulations. These environments, combined with Gazebo's enhanced physics and realistic rendering of lighting and materials, contribute to a more realistic user experience and simulation realism. Gazebo's latest version, called Gazebo Garden, is the version that we are using for our implementation. Figure 4 shows an example scene simulated using Gazebo garden.



World Sdf	
+ Physics	
Name	shapes
Atmosphere	
Physics Engine Plugin	
Physics Collision Detector	ode
+ Magnetic Field	
Render Engine Gui Plugin	
Render Engine Server Plugin	
Physics Solver	DantzigBoxedLcpSolver
Scene	
Render Engine Server Headless	
+ Gravity	
+ System Plugin Info	

Figure 4: Example of simulation environment in Gazebo Garden.

PX4 is an open-source flight control software for UAVs and other unmanned vehicles. It provides a high-level interface to control vehicle navigation and flight, and it supports a wide range of vehicle designs. The project is backed by a strong community of developers and industry partners, and it is used in a variety of applications, from hobbyist flying to commercial UAV operations. Software In The Loop (SITL)

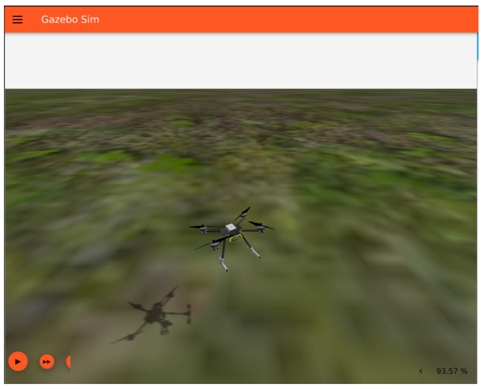


Figure 5: A PX4 SITL simulation in Gazebo garden

simulation is a method for testing software in a realistic environment without the risk or expense of a real vehicle. PX4 SITL allows developers to run the entire flight stack, including the state estimator, control loops, and higher-level modules, on a standard desktop computer. This makes it possible to test the software in a wide range of scenarios without ever leaving the lab. Figure 5 shows an example of a PX4 SITL simulation.

Gazebo's realistic simulation environments, combined with PX4's robust flight control software, provides an excellent platform for running UAV simulations. This setup can simulate a wide range of scenarios, from basic flight control to complex navigation tasks, in a safe and controlled environment. It also allows for rapid testing and iteration in a safe and controlled environment, reducing the risk and cost associated with testing on physical UAVs. This framework is particularly useful for testing object-avoidance, computer vision, and multi-vehicle coordination algorithms. It also supports continuous integration tests, making it an essential tool for modern UAV software development workflows. The use of open-source tools like Gazebo and PX4 ensures that our research is reproducible and can be easily built upon by other researchers in the field.

3.1.2 ROS2 middle-ware and the gz-transport module

The Robot Operating System (ROS) is a set of software libraries and tools for building robot applications. ROS2 is the second version of this system and offers several improvements over its predecessor. It provides the means to control a wide range of vehicle and robot designs and is used in a variety of applications. At its core, ROS2 provides the necessary services for hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS2 uses a graph architecture to manage the complexity of robotic systems. In this architecture, processing takes place in nodes that may receive and post sensor, control, state, planning, actuator and other messages. These nodes are loosely coupled using the ROS communication infrastructure and can be distributed over a network.

The gz-transport module is a part of the Gazebo simulator. It is a transport library that allows for the efficient exchange of data between different parts of a Gazebo simulation. This exchange includes sensor data, control commands, and other information that is crucial for the operation of a simulated robot. The gz-transport module is designed to be fast and efficient, enabling real-time communication between different parts of a simulation. This feature makes it an essential tool for complex simulations involving multiple interacting robots or other elements. The combination of ROS2 and the gz-transport module provides a powerful toolset for developing and testing UAV software. ROS2 offers high-level control capabilities, while the gz-transport module ensures efficient data exchange within the simulation.

The integration of the ROS2 framework for high-level control of the PX4 UAV has yielded commendable results. However, the ROS2 Gazebo bridge is not optimal, as it was observed to reduce the Real-Time Factor (RTF) in our simulations. The Real-Time Factor (RTF) is a metric indicating the simulation's execution speed relative to real time. It measures the time passed in the simulation compared to the real world. The gz-transport module, on the other hand, allows for efficient data exchange within the simulation, improving speed by eliminating an additional middleware layer.

In our experiment, we utilized the gz-transport module for low-level control tasks such as subscribing to the animals' positions, publishing velocity commands to the animals, subscribing to the UAV's IMU data, and publishing the joint angles of the gimbal mounted on the UAV. On the other hand, we employed ROS2 for high-level UAV commands, which include setting position and velocity setpoints for different herding algorithms, publishing commands to arm, take off, and land the UAV, among others. However, for retrieving images from the UAV's camera, we used ROS2 instead of the gz-transport module due to its relative ease of use in this context.

3.1.3 Quadcopter and the gimbal cam setup

The quadrotor utilized in the Software In The Loop (SITL) simulation is the x500 UAV. This UAV is a popular choice among UAV developers due to its robust design and compatibility with the PX4 development kit. The SDF (Simulation Description Format) model of the x500 UAV is employed in the simulation, making the solution easily deployable in real-world hardware. With minor modifications, this can include the Holybro x500 v2 PX4 development kit, as shown in Figure 6.

However, the SITL version of the x500 UAV does not include a default camera mount with a gimbal. To overcome this, we manually added a camera to the UAV along with a gimbal mount. This addition was achieved by modifying the SDF file and controlling the gimbal axes positions using the Joint Position Controller Plugin. The Joint Position Controller Plugin in Gazebo is a tool that allows for the precise control of joints in a simulated robot. We used this plugin to send target joint positions to a robot in simulation through a simple joint position command ROS topic publisher or a gz-transport publisher. It can be employed in an SDF model spawned in a Gazebo simulation. Figure 7 illustrates the gimbal we implemented.

In the context of the x500 UAV, the Joint Position Controller Plugin is used to control the gimbal axes positions. This allows for precise control of the camera's

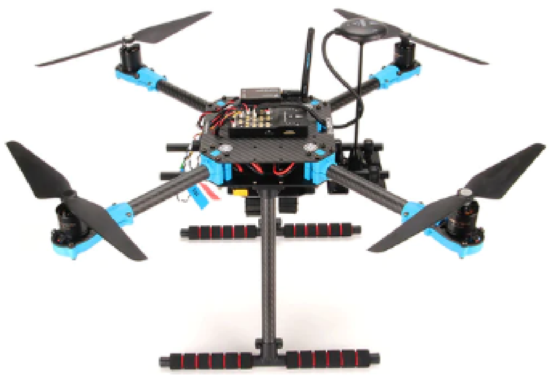
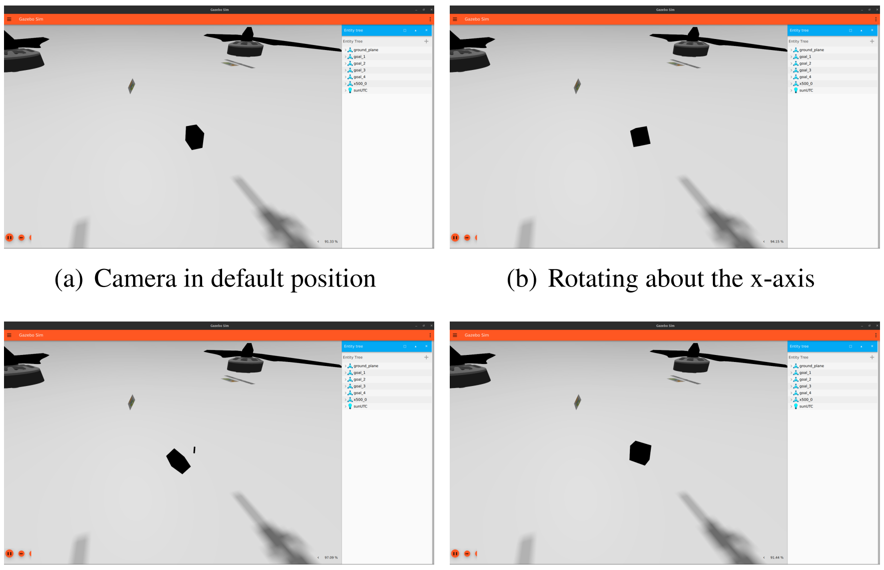


Figure 6: The X500 UAV setup with the PX4 flight controller



(a) Camera in default position

(b) Rotating about the x-axis

(c) Rotating about the y-axis

(d) Rotating about the z-axis

Figure 7: Implementation of a gimbal in Gazebo garden

orientation, enabling it to maintain a steady view of the ground below, irrespective of the UAV's movements. The gimbal is set to point at a 45-degree angle downwards. It maintains this orientation with respect to the local/global frame by counteracting the effect of the UAV's orientation. This setup enables efficient tracking and herding of the animals in the simulation, demonstrating the potential of this technology for real-world applications.

The following formula is used to calculate the pitch angle of a camera gimbal mounted on a UAV, ensuring that the camera maintains a 45-degree downward tilt relative to the local frame, regardless of the UAV's pitch.

$$\theta_g = \frac{\pi}{4} - \theta_d$$

In this context, θ_g denotes the pitch angle of the camera gimbal, θ_d represents the pitch angle of the UAV, and $\pi/4$ signifies the desired downward tilt of the camera in radians. The aim of this formula is to maintain a consistent 45-degree downward tilt for the camera with respect to the local frame, irrespective of the UAV's pitch. When the UAV pitches its nose upward (indicated by a negative θ_d), the gimbal's pitch angle (θ_g) increases to a higher positive value, inducing a nose-down pitch. This action effectively counteracts the UAV's pitch, enabling the camera to maintain its 45-degree downward tilt relative to the local frame, rather than the UAV's body frame.

3.1.4 Simulating the animals

In the given experiment, the animals are represented by boxes with ArUco markers on each side. ArUco markers are synthetic square markers composed of a wide black border and an internal binary matrix, which determines their identifier (id). The black border facilitates their fast detection in an image, and the binary codification, consisting of black and white colors, allows for identification and the application of error detection and correction techniques. These markers are often enclosed within an additional white outer border to improve detection accuracy. They are commonly used in pose estimation, camera calibration, robot navigation, and augmented reality applications. The simulation setup can be seen in Figure 8.

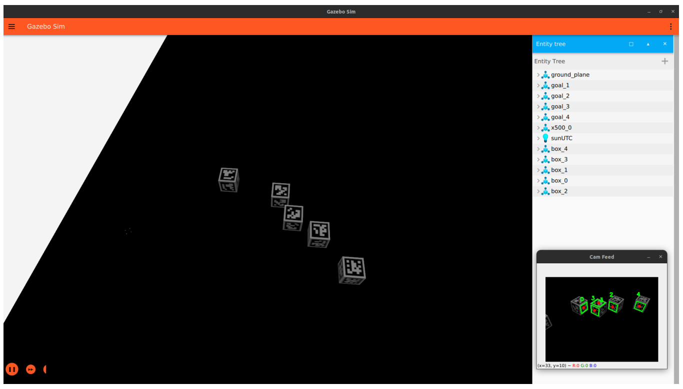


Figure 8: The experimental setup

The contrast between the black floor and the white outer borders of the ArUco markers on the boxes facilitates the camera's detection and identification of the markers. The boxes, representing the animals, can move in a 2D plane, simulating the movement of animals on a flat surface. This movement can be controlled to simulate various behaviors and scenarios. The boxes can be spawned in random positions, adding an element of unpredictability to the simulation and increasing the challenge for the UAV to herd the animals.

The use of ArUco markers in this simulation offers a simple and effective way to track the position and movement of the animals. Although animals in the real world need to be detected using sophisticated computer vision algorithms for object detection, such as YOLO Object Detection [32], this setup acts as a commendable substitute for the real world. The markers can be easily detected and identified by the UAV's camera, enabling the UAV to accurately track each animal's position and adjust its herding strategy accordingly. This setup eliminates the need to implement sophisticated trajectory estimation and tracking algorithms to keep track of all the unique animals detected by the UAV's camera. Instead, each animal having a unique marker makes it easier for the herding algorithm to detect the prolonged absence of certain animals when their ArUco marker is not detected by the UAV's camera.

3.2 Implementing the herding algorithm

3.2.1 Simulating the Flocking Behaviour

The behaviour of the animals, which are represented as markers, is governed by a set of rules simulating flocking behavior. These rules are implemented as mathematical equations determining the animals' movements based on their positions relative to each other and the predator (the UAV). This experiment uses the same rules used by Barksten et al. [5] in their experiment to simulate the behaviour of sheep in the presence of a human-controlled predator, with minor modifications to suit our experiment. The rationale behind using these rules is that the objects with ArUco markers exhibit flocking behaviour akin to the behaviour of actual animals present in the real world, as demonstrated by the work of Barksten et al. [5].

- **Cohesion** - The cohesion rule is used to simulate the tendency of animals in a flock to stay close to each other while moving. This rule calculates a vector directed towards the Center of Mass (COM) of all the animals. Mathematically, this is represented as:

$$\mathrm{coh}(s) = \frac{S_p - s_p}{|S_p - s_p|} \tag{1}$$

  where S_p is the average position of all the animals and s_p is the position of the current animal.

- **Separation** - The separation rule prevents animals from colliding with each other by directing them away from nearby animals. The rule calculates a vector directed away from each nearby animal, with the contribution of each animal determined by an inverse square function of the distance between the animals. This is represented as:

$$\mathrm{sep}(s) = \sum_{i=1}^{n} \left(\frac{s_p - s_{ip}}{|s_p - s_{ip}|} \cdot \mathrm{inv}(|s_p - s_{ip}|, 2) \right) \tag{2}$$

  where s_{ip} is the position of the i-th nearby animal. For simplicity, our experiment considers all the animals present in the simulation as nearby animals. In this equation, $\mathrm{inv}(|s_p - s_{ip}|, 2)$ is a function of the form $\mathrm{inv}(x, s)$. This function is used to prioritize nearby objects in the separation and escape rules. It is an inverse square function of the distance x between the objects, with a softness factor s to slow down the rapid decrease of the function value. We found that the softness factor of 2 works better than the original softness factor of 1 used in [5]. The inverse square function is given by the equation:

$$\text{inv}(x, s) = \left(\frac{x}{s} + \varepsilon\right)^{-2} \tag{3}$$

in this equation, ϵ is a small numerical value that prevents a division by zero error, when x assumes the value of 0.

- **Alignment** - The alignment rule makes animals move in the same direction as their nearby flockmates. The rule calculates a vector directed in the average direction of all nearby animals. This is represented as:

$$\text{ali}(s) = \frac{1}{N} \sum_{si \in N} s_{iv} \tag{4}$$

  where s_{iv} is the normalized velocity of the i-th nearby animal and N is the total number of nearby animals. This means that s_{iv} is a unit vector pointing in the direction where the i-th nearby animal is headed. For the alignment rule, animals that are within 10 meters of the current animal in the simulation are considered nearby animals. The alignment equation shown above varies slightly from the alignment equation used in [5], as it was observed that the magnitude of the velocities exploded when the sum of the velocities was used instead of the average of all the velocities.

- **Escape** - The escape rule makes animals flee from the predator. The rule calculates a vector directed away from the predator, with the size of the vector determined by an inverse square function of the distance between the animal and the predator. This is represented as:

$$\text{esc}(s) = \frac{s_p - p_p}{|s_p - p_p|} \cdot \text{inv}(|s_p - p_p|, 20) \tag{5}$$

  where p_p is the position of the predator. We found that a softness factor of 20 in the inverse function is more suitable than the softness factor of 10 used in [5].

It is required to combine all these rules to mimick the behaviour of animals in the presence of a predator agent. The final velocity vector v of the animal is the weighted sum of all the rule vectors, with each rule weighted by a combined multiplier. The combined multiplier is a function used to combine the two multipliers for each rule. It is of the form:

$$m(1 + p(x)m_p) \tag{6}$$

Where m is the first multiplier and $p(x)$ is the second multiplier scaled by a value mp. The first multiplier m and the scaling factor m_p are constants, while the second multiplier $p(x)$ is a function of the distance x from the animal to the predator given by:

$$p(x) = \frac{1}{\pi} \arctan\left(3(r_{flight} - x)\right) + 0.5 \tag{7}$$

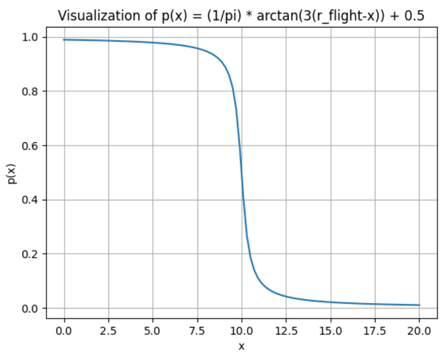


Figure 9: Mathematical visualisation of the second multiplier function.

Where r_{flight} is the flight zone radius of the animal or the distance at which the animals start noticing the presence of an approaching predator. Figure 9 provides an illustration of the second multiplier function for $r_{flight} = 10$. When the predator is beyond a distance of 10 meters from the animals, the effect of the multiplier $p(x)$ is close to zero. While the effect of $p(x)$ is close to 1 when the predator is within a distance of 10 meters from the flock. This results in a behaviour that mimics the effect where the animals would sense the presence of a predator and alter their behaviour when it approaches them by amplifying their cohesion, separation and alignment properties.

Each animal's final velocity vector is the weighted The resultant direction of each animal's movement is determined by the sum of all the rule vectors, with each rule weighted by a combined multiplier. The vector is capped at a maximum velocity v_{max} and a minimum velocity v_{min}, which, in our case, is the negative of v_{max}. Clipping is performed to ensure that the animals' velocity doesn't explode to infinity when the UAV flies over the flock to get to the other side. This is represented as:

$$\begin{aligned} v =& m_c(1 + p(x)m_{cp})\text{coh}(s) \\ &+ m_s(1 + p(x)m_{sp})\text{sep}(s) \\ &+ m_a(1 + p(x)m_{ap})\text{ali}(s) \\ &+ m_e\text{esc}(s) \end{aligned} \tag{8}$$

$$v = (0.8 * v) + (0.2 * v_{prev}) \tag{9}$$

$$v_{current} = max(min(v, v_{max}), v_{min}) \tag{10}$$

$$v_{prev} = v_{current} \tag{11}$$

where m_c, m_s, m_a, and m_e are the weights for the cohesion, separation, alignment, and escape rules respectively, and m_{cp}, m_{sp} and m_{ap} are the weights for the second multipliers for the first three rules respectively. The sum v is weighted and added to a weighted value of the previous velocity value v_{prev} to simulate inertia. Finally, this value is clipped by v_{min} and v_{max}. The resultant effect is the behavior of boxes flocking together in the absence of a predator and moving away from the predator, spreading outwards when a predator approaches the animals.

Implementation of this flocking behavior in the presence of a UAV with a gimballed camera sets up the environment required to employ different herding algorithms and compare their performances.

3.2.2 Baseline Algorithm 1: The Collect and Drive Algorithm

For our comparative analysis, we employ the Collect and Drive (CD) algorithm, a heuristic-based strategy proposed by Strömbom et al. [6], that simulates a shepherd's herding behavior. One of the reasons for implementing this algorithm is the ease of implementing a computer vision stack on top of this algorithm, as it uses the position of the UAV and the animals to predict the target setpoint for the UAV. Despite its simplicity, this algorithm achieves excellent performance as observed in the experiments conducted by Strömbom et al. [6]. The CD algorithm operates in two primary modes: Collect and Drive. In the Collect mode, the shepherd (UAV) gathers the dispersed flock, and in the Drive mode, it guides the aggregated group towards the target location. The detailed pseudo code for the Collect and Drive algorithm is presented in Algorithm 1.

In this pseudo code, the shepherd (UAV) dynamically switches between the Collect and Drive modes based on the dispersion of the flock. Note that the UAV always starts in the Collect mode. The shepherd focuses on the local centre of mass of nearby agents, allowing for more effective control of the group. Here, C is the mean position of the flock, G is the goal position, F is the position of the furthest animal, and D is the UAV's target position setpoint. $d_{\min}$ is the minimum distance required between the UAV and its target entity, either the center of the flock or the furthest agent, and θ is the UAV's yaw angle. The function arctan calculates the angle at which the UAV must rotate to point towards the mean position of the flock. The UAV's position and yaw angle are updated at each step based on the current mode and the state of the flock.

3.2.3 Baseline Algorithm 2: The Farthest Agent Targeting Algorithm

The second baseline algorithm we will be comparing our proposed algorithm against is the Farthest Agent Targeting algorithm proposed by Tsunoda et al. [22]. This algorithm operates on the principle of the shepherd UAV targeting the sheep that is farthest from the goal, thereby ensuring that the most 'problematic' agent is guided towards the goal. This strategy allows the shepherd to gradually gather all the sheep into the goal zone. The main rationale for choosing this algorithm is its proven ability to herd loosely gathered flocks of animals using the local information obtained from a camera sensor mounted on the UAV, as observed in the study proposed by Tsunoda et al. [22]. The pseudo code for the Farthest Agent Targeting algorithm is presented in Algorithm 2.

Algorithm 1 Collect and Drive Algorithm

```
Initialise: $collect = True$, $r$ = flock radius, $N$ = number of sheep, $r_a$ = agent to
agent interaction distance, $d_{min}$ = minimum required distance between the UAV and
the target.
while not at the goal do
    if $collect = False$ then
        if $r \leq r_a N^{3/4}$ then
            $v_{\text{drive}} = (C - G)/|C - G|$.
            $D = v_{\text{drive}} \cdot d_{\text{min}} + C$.
        else
            $collect = True$.
        end if
    else
        if $r \geq r_a N^{1/3}$ then
            $v_{\text{collect}} = (F - C)/|F - C|$.
            $D = v_{\text{collect}} \cdot d_{\text{min}} + F$.
        else
            $collect = False$.
        end if
    end if
    $\theta = \arctan(C_y - D_y, C_x - D_x)$.
end while
```

Algorithm 2 Farthest Agent Targeting Algorithm

```
Initialise: $K_1$, $K_2$, $K_3$, $d_{min}$ = minimum required distance between the UAV and the
target.
while not at the goal do
    $u_1 = \frac{(T_i - D)}{|T_i - D|}$.
    $u_2 = -\frac{(D - T_i)}{|D - T_i|^3}$.
    $u_3 = -\frac{(D - G)}{|D - G|}$.
    $u = K_1 \cdot u_1 + K_2 \cdot u_2 + K_3 \cdot u_3$.
    Set UAV's velocity setpoint as $u$ with some random noise.
    $\theta = \arctan(C_y - D_y, C_x - D_x)$.
    Set UAV's yaw as $\theta$.
end while
```

In this algorithm, three vectors are calculated: u_1, u_2, and u_3. The vector u_1 points from the UAV towards the farthest agent, attracting the UAV to this agent. The vector u_2 points from the farthest agent towards the UAV, repelling the UAV from getting too close to the agent. The vector u_3 points from the goal position to the UAV, repelling the UAV away from the goal. T_i is the position of the farthest agent, D is the UAV's position, G is the goal position, and C is the position of the flock's center. The UAV's velocity is then set as the sum of these three vectors, each multiplied by their respective coefficients K_1, K_2, and K_3 along with some noise to avoid getting trapped in an equilibrium achieved by the vectors u_1, u_2, and u_3 cancelling each other out. The UAV's yaw is set to point towards the center of the flock. This approach ensures that the UAV effectively guides the farthest agent towards the goal, thereby managing the flock efficiently.

In the subsequent sections, we will elaborate on the different components of our proposed algorithm.

3.3 Our Proposed Modifications

3.3.1 Patrolling and Transition to Herding

In the realm of autonomous UAV herding, the initial phase of operation is of paramount importance. This phase, known as 'patrolling', involves the UAV navigating through a series of predefined waypoints scattered across the environment. The primary objective of this patrolling phase is to scout for animals and assess the need for herding.

Unlike the baseline algorithms, which operate under the assumption of prior knowledge about the animals' positions, our approach embraces a more realistic scenario. In real-world applications, especially when a single UAV is deployed for herding, the initial positions of the animals are typically unknown. This necessitates the patrolling phase, where the UAV embarks on an exploratory journey to locate the animals. The patrolling algorithm is explained in Algorithm 3

Algorithm 3 Patrolling Algorithm

```
Initialise a set of predefined waypoints $W$.
Initialise a counter $i = 0$.
while not in herding mode do
    Move the UAV to the waypoint $W[i]$.
    if at least two unique animals are detected then
        Switch to herding mode.
    else
        $i := (i + 1) \mod |W|$.
    end if
end while
```

Where $\mod |W|$ is the modulus operator which returns the reminder when dividing $(i + 1)$ by the length of the set W. This algorithm represents a simple patrolling strategy where the UAV cyclically moves between a set of predefined waypoints until it detects at least two unique animals, at which point it switches to herding mode. This is to ensure the reliability of detection and to confirm the existence of a group of animals. The waypoints are scattered all over the environment in such a way that the UAV can scout the entire farm during the patrolling process. The counter *"i"* is used to cycle through the waypoints in order.

In summary, the patrolling phase is a crucial preparatory step in our algorithm. It enables the UAV to gather essential information about the environment and the animals, setting the stage for the subsequent herding phase.

3.3.2 The Position Update Algorithm

The Position Update Algorithm is a key component of our herding strategy, designed to keep track of the animals' positions in real-time. The algorithm operates by maintaining a dynamic list, which is updated as the UAV detects more unique animals. The position update algorithm is described in Algorithm 4.

Algorithm 4 Position Update Algorithm

```
Initialise an empty set $D$ for the detected animals and empty function $L$ that maps
each detected animal to its last known position.
$D := \emptyset$
$L := \emptyset$
for each detected animal $a$ do
    if $a \notin D$ then
        $D := D \cup \{a\}$
        $L(a) := \text{position}(a) + N(0, \sigma)$
    else
        $L(a) := \text{position}(a) + N(0, \sigma)$
    end if
end for
```

Here, position(a) represents the real-time position of the animal a, which can be subscribed to using the corresponding topic at any point during the simulation. However, the algorithm accesses the real-time position only when the animal is detected in the current frame. The function $N(0, \sigma)$ represents a Gaussian noise with mean 0 and standard deviation σ, which is added to the original position of the detected animals to simulate the operation of real-world position triangulation algorithms such as the algorithm proposed by Zhao et al. [16] for performing camera based position triangulation.

In essence, the Position Update Algorithm provides a systematic approach to track the animals' positions, enabling the UAV to effectively manage the herding process. In the event of a sheep not being detected for a long time it becomes the farthest agent from the target or the sheep which is farthest away from the center of the flock, so the UAV takes the appropriate action based on whether the herding algorithm involved is FAT or CD. One drawback of this algorithm is that it depends on the reliability of the object detection algorithm, which detects the unique marker. Failure of the detection algorithm to be robust would cause the herding algorithm to fail.

3.3.3 Proposed Vision-Based Algorithms

In our study, we present vision-based extensions applicable to both the Collect and Drive (CD) and Farthest Agent Targeting (FAT) algorithms. These extensions build upon the baseline concept, introducing two enhanced approaches. The proposed algorithms incorporate the use of a computer vision pipeline to estimate the positions of the animals, thereby making the herding process more realistic and applicable to real-world scenarios.

The first proposed algorithm, referred to as the CD Vision algorithm, is an extension of the baseline CD algorithm. The key difference lies in the source of the animal positions used in the algorithm. Instead of using real-time positions of the animals, the CD Vision algorithm utilizes the noisy position estimates of the animals, which are computed using the position update algorithm. This approach allows the algorithm to operate under the realistic assumption that the exact positions of the animals are not always known.

Similarly, the second proposed algorithm, known as the FAT Vision algorithm, extends the baseline FAT algorithm. Like the CD Vision algorithm, the FAT Vision algorithm uses the noisy position estimates of the animals, computed using the position update algorithm. This allows the FAT Vision algorithm to target the "problematic" agent that is farthest from the goal, even when the exact positions of the animals are not known.

By incorporating the use of a computer vision pipeline, these proposed algorithms offer a more realistic approach to herding. They do not rely on the assumption that the exact positions of the animals are known, making them more applicable to real-world herding scenarios where such information may not always be available.

3.3.4 Establishing the Termination Condition

To ensure the UAV's operation is finite and purposeful, it is crucial to define a precise termination condition. This condition signifies the completion of the herding task and prompts the UAV to return to its original launch point, also known as the home position.

In the context of our algorithm, the termination condition is met when the geometric center of the flock is within a specified proximity threshold of the designated goal position. Once this condition is satisfied, the herding task is deemed successful, and the UAV is instructed to return to its home position.

If the UAV is not able to herd the animals within 5 minutes (300 seconds) or if the UAV doesn't spot the animals continuously for 30 seconds, the herding is deemed a failure, and the UAV is instructed to return to its home position.

It is worth noting that the selection of the goal position plays a significant role in the effectiveness of the herding operation. The goal position should be chosen strategically to prevent the animals from immediately reverting to their initial location post-herding. This careful selection ensures that the herding algorithm fulfills its purpose and achieves the desired outcome.

3.3.5 The UAV Herding Pipeline

The proposed herding pipeline is depicted in Figure 10. The herding process begins with the UAV taking off and cycling between different waypoints during the patrolling mode. Once the UAV detects two distinct animals, it transitions to the herding mode

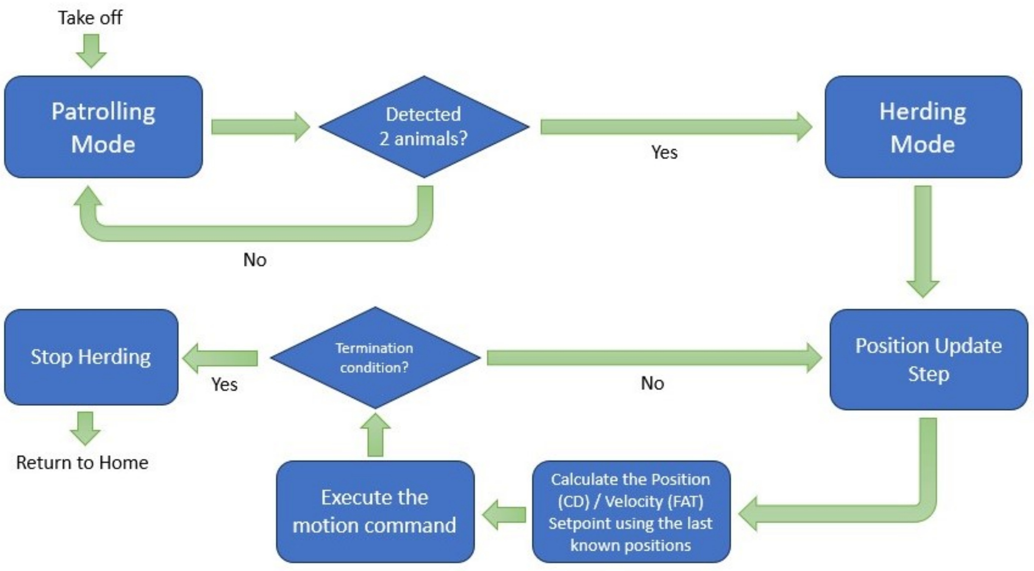


Figure 10: The herding pipeline

described in Algorithm 3. Two herding algorithms, namely the CD algorithm and the FAT algorithm, can be utilized.

During the herding phase, at each time step, the camera captures a live image and performs ArUco marker detection. If any ArUco markers are detected, the positions of the animals are updated based on Algorithm 4. The updated positions of the detected animals are then used as input for the herding algorithm, instead of using the actual positions of the animals.

The herding algorithm calculates and publishes the position setpoint (for the CD algorithm) or velocity setpoint (for the FAT algorithm) to the UAV's control node using ROS2. The termination condition is checked to determine if the herding process has succeeded or failed. If the termination condition is met, indicating a successful or unsuccessful herding, the UAV returns to the home position and lands.

Otherwise, the UAV repeats the position update step with the latest image captured by the camera and continues the subsequent steps in a loop until the herding process terminates. Hence, the proposed algorithm enhances the existing CD and FAT-based algorithms by incorporating a patrolling algorithm and a position update algorithm into the pipeline. This approach enables the testing of herding algorithms on a UAV in a three-dimensional environment using computer vision as the primary sensing modality. It incorporates features such as patrolling and automatic return to home once the herding process concludes, regardless of its success or failure.

4 Results

Building upon the methodology outlined in the previous sections, we will now delve into the results of our study. We evaluated the performance of two categories of herding algorithms: the baseline algorithms and the vision-based algorithms, each having two variants: the Collect and Drive (CD) algorithm and the Furthest Agent Targeting (FAT) algorithm. The baseline algorithms assume perfect knowledge of the positions of all animals, whereas the vision-based algorithms utilize computer vision to determine the animal positions.

4.1 Experimental Setup

The experimental setup is designed to simulate a realistic environment in which a UAV is employed for animal herding. The setup outlines the metrics that will be tested and the parameters that will be utilized for the experiments.

4.1.1 Metrics to be analysed

The following metrics will be utilized to assess the performance of our proposed herding algorithm. A comparison will be made between the metrics of the proposed algorithm and those of the baseline algorithms to evaluate the effectiveness of the proposed vision algorithm.

- **Flock Radius:** This refers to the distance between the center of a flock and the farthest animal from the center. A robust algorithm aims to minimize this value during the herding process.
- **Herding Time:** This is the duration taken by the herding algorithm to guide the animals towards the target location. An effective algorithm accomplishes herding in a shorter time.
- **Success Rate:** This indicates the percentage of times a herding algorithm successfully herds a group of animals within a specified time frame without losing track of the animals, based on a total of n test runs. An ideal algorithm achieves a 100 percent success rate.
- **Number of Animals Successfully Herded:** This represents the count of animals that the UAV is able to herd with a high success rate before a decline in performance occurs. A larger number signifies a more robust algorithm.
- **Distance Travelled:** This denotes the total distance covered by the UAV to complete the herding task. A shorter distance traveled implies reduced UAV movement, lower battery consumption, and a more efficient algorithm.

4.1.2 UAV and Herding Algorithm Parameters

- In the CD algorithm, the UAV maintains a minimum distance of 13 meters from its target entity, regardless of the number of animals and their behavior parameters.
- In the CD algorithm, the UAV switches between collect and drive modes based on the flock radius (r), agent-to-agent interaction distance (r_a), and the number of sheep (N):
  - It switches to collect mode when $r > r_{collect_N}$, where $r_{collect_N} = r_a N^{3/4}$.
  - It switches to drive mode when $r < r_{drive_N}$, where $r_{drive_N} = r_a N^{1/3}$.
- In the FAT algorithm, the coefficients for u_1, u_2, and u_3 are set as $K_1 = 9.0$, $K_2 = 0.15$, and $K_3 = 3.0$, respectively.
- The flight radius, which represents the distance at which the animals feel threatened by an approaching UAV, is set at 15 meters for all the following experiments.
- Similarly, the UAV always strives to maintain an altitude of 14 meters from the ground.

4.1.3 Flocking Parameters

The agent-to-agent interaction distance (r_a) is calculated using the separation coefficients (m_s and m_{sp}), cohesion coefficients (m_c and m_{cp}), and the escape rule coefficient (m_e). The formula is as follows:

$$r_a = (m_s + m_e - m_c) + \frac{(m_{sp} - m_{cp})}{30} \tag{12}$$

The interaction distance increases when the escape force and separation forces are high, and decreases when the cohesive forces are high. The coefficients m_{sp} and m_{cp} come into play only when the predator (UAV) is in close proximity, which is why they are divided by 30.

4.1.4 Simulation Conditions

The following information is necessary to replicate our simulation experiment:

- Animals are randomly spawned with x and y coordinates ranging from +10m to +50m, as depicted in Figure 12.
- The UAV herds these animals to the bottom-left quadrant (-40,-40).
- The initial patrolling waypoint is positioned closer to the animals' spawn location to expedite the simulation, although this can be modified.

- Herding is considered successful when the center of mass of the animals is within a 5-meter radius of the goal position.
- Herding is deemed a failure if it exceeds 5 minutes on the simulation clock or if no animals are detected for a continuous period of 30 seconds.
- We have assumed a position update algorithm with zero noise, thus conducting the herding study in a near-ideal environment.

To assess the algorithms' robustness, we will conduct herding tests in three distinct scenarios, each with different flocking parameters. Specifically, we will evaluate each scenario with 5, 10, and 15 animals. For each combination of herding algorithm, number of animals, and flocking parameters, we will perform 10 simulations to ensure reliable and accurate results. By analyzing the average performance across the 10 attempts, we can gain a comprehensive understanding of the algorithms' robustness under various scenarios and animal population sizes. To test the true capabilities of these algorithms, more testing in a wide variety of scenario with the UAV and animals being spawned at more random location is required. However, given that ever run of the herding simulation can take up to 5 minutes, simulating all the possibilities would take a long time. Thus, the study is limited to a small subset of the possible scenarios.

4.2 Herding Performance Analysis

In this section, we analyze the results of our experiments conducted under three distinct scenarios, each representing a different animal behavior profile characterized by specific flocking parameters. The performance of the herding algorithms is evaluated in each scenario.

4.2.1 Scenario 1: Closely Packed, Fast Animals

This scenario features animals that are relatively easier to herd due to their tendency to remain closely packed and their ability to move quickly when required. The flocking parameters for this scenario are as follows:

$m_c = 2.0, \quad m_s = 4.5, \quad m_a = 0.0, \quad m_e = 3.0, \quad m_{cp} = 4.5, \quad m_{sp} = 14.5,$
$m_{ap} = 10.0, \quad v_{max} = 50.0, \quad v_{min} = -50, \quad r_a = 5.833, \quad r_{collect_5} = 19.50,$
$r_{collect_{10}} = 32.80, \quad r_{collect_{15}} = 44.46, \quad r_{drive_5} = 9.97, \quad r_{drive_{10}} = 12.57,$
$r_{drive_{15}} = 14.39$

The above parameters result in a flock that behaves in a way illustrated by Figure 11. The reason for this is that the cohesive force is higher due to higher coefficient m_c while the separation force is only slightly greater since m_s is slightly higher compared to the other scenarios. The escape coefficient (m_e) of 3 has a significant impact on the repellence of the animals. The multiplier rule coefficients m_{cp}, m_{sp} and m_{ap} have a more pronounced effect on herding when the predator approaches the flock due to equation 8. The animals are thus repelled from each other with a higher magnitude when the predator approaches the flock which is observed later in Figure 13, when the experiments are conducted. The wide margin between v_{max} and v_{min} make it possible for the animals to run as fast as possible from the predator if required. The value of r_a for this scenario is the result of these flocking rule coefficients which is lesser than the r_a value of the other scenarios. r_a is calculated using equation 12. The values of $r_{collect_5}$, $r_{collect_{10}}$, $r_{collect_{15}}$, r_{drive_5}, $r_{drive_{10}}$, $r_{drive_{15}}$ determine the radius at which the collect and drive algorithm switches between collect and drive mode for scenario 1 for the three possible animal population sizes selected. They are determined as shown in Section 4.1.2. The gap between the thresholds for a particular population size is smaller compared to the other scenarios which will be discussed later.

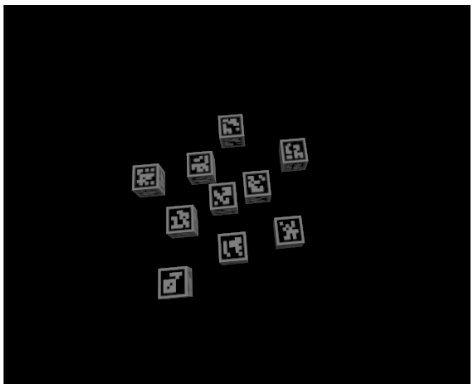


Figure 11: Scenario 1: Closely Packed Animals

The observations made during the simulation with the aforementioned parameters are as follows:

1. **Herding Success Rate:** The success rate of herding for Scenario 1 was recorded over 10 repetitions for each herding algorithm across different animal population sizes. The results are presented in Table 1.

	Animal Population		
	5 Animals	**10 Animals**	**15 Animals**
CD Baseline	100%	100%	100%
FAT Baseline	100%	100%	100%
CD Vision	90%	0%	0%
FAT Vision	90%	20%	0%

Table 1: Herding Performance for scenario 1.

Based on the data, we can observe that the CD Baseline and FAT Baseline algorithms consistently achieved a 100% herding success rate across all animal population sizes. However, the CD Vision and FAT Vision algorithms exhibited a decrease in effectiveness as the animal population increased. These vision-based algorithms faced challenges in maintaining successful herding, particularly with larger animal populations. Furthermore, the FAT Vision algorithm demonstrated slightly better performance compared to the CD Vision algorithm for larger population sizes.

2. **Average herding Time:** The average herding time for each algorithm is presented in Table 2. Empty entries in the table indicate that a particular algorithm failed to achieve successful herding in any of the 10 attempts for that specific situation. It is important to note that the time measured here corresponds to the simulation clock time rather than real-time. The real-time taken to complete the simulation may vary based on the computer's processing capability. Therefore, measuring the simulation time provides a consistent metric across different devices.

	Animal Population		
	5 Animals	**10 Animals**	**15 Animals**
CD Baseline	61.38	101.79	157.42
FAT Baseline	43.06	78.24	100.50
CD Vision	62.43	-	-
FAT Vision	49.828	83.78	-

Table 2: Average herding time in seconds for scenario 1

Based on the data in the table, we can observe that the CD Baseline algorithm generally has longer average herding times compared to the FAT Baseline algorithm across all animal population sizes. The CD Vision algorithm, although successful only for 5 animals, exhibits a similar average herding time compared to the CD Baseline algorithm for that population size. Similarly, the FAT Vision algorithm, successful for 5 animals and 10 animals, demonstrates average herding times comparable to the FAT Baseline algorithm for those respective population sizes. These findings suggest that the FAT Vision algorithm outperforms both the CD Vision algorithm and the CD Baseline algorithm in terms of average herding time for successful scenarios.

3. **Average Distance Traveled:** The table 3 displays the average distance traveled by the UAV during the herding process.

	Animal Population		
	5 Animals	**10 Animals**	**15 Animals**
CD Baseline	336.74	485.94	215.11
FAT Baseline	342.69	594.19	768.81
CD Vision	366.90	-	-
FAT Vision	365.57	527.83	-

Table 3: Average distance travelled in meters for scenario 1

Based on the available data, it is evident that both the CD Baseline and FAT Baseline algorithms exhibit an increase in the average distance traveled as the animal population size increases. The FAT Baseline algorithm consistently covers a greater distance compared to the CD Baseline algorithm across all population sizes. In the successful scenario, the CD Vision algorithm covers a slightly greater distance than the CD Baseline algorithm. Similarly, the FAT Vision algorithm covers a similar distance to the FAT Baseline algorithm. However, due to the lower success percentage and limited data, it is challenging to draw a definitive conclusion regarding the performance of the FAT Vision algorithm compared to the Baseline algorithm.

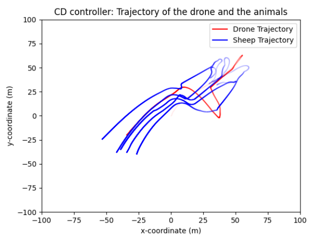
4. **Herding Trajectory:** The herding trajectories of the baseline algorithms and the vision-based algorithms, tested on scenario 1 with different numbers of animals, are depicted in Figure 12. Each trajectory is unique for each attempt, but the presented graphs illustrate the most common observed pattern for the tested situations.

   In scenario 1, the baseline algorithms demonstrate effective herding performance across different animal population sizes. However, the performance of the vision-based algorithms is most pronounced when there are five animals present, as their effectiveness decreases with an increase in the number of animals. Notably, the FAT Vision algorithm exhibits a similar smooth trajectory as the FAT Baseline algorithm in the case of five animals, indicating its ability to consistently track all the animals without losing sight of them for an extended period.

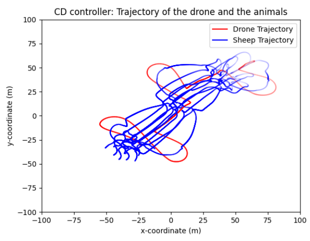
5. **Flock Radius:** The change in the actual radius of the flock over time when the baseline and the vision based algorithms are deployed in scenario 1 for different animal population sizes are shown in Figure 13. These graphs correspond to the herding attempts in the herding trajectory graphs shown above.

   In this analysis, the behavior of the CD baseline algorithm is characterized by frequent switches between collect and drive modes in response to changes in the flock radius, except for the scenario with a population size of 15. This exceptional behavior is attributed to the requirement of a flock radius less than $r_{drive_{15}} = 14.39$ meters for the algorithm to transition into drive mode in a loosely packed scenario. Given that the algorithm always starts in collect mode, the UAV never transitioned into drive mode and instead attempted to bring the animals together while staying in collect mode. Therefore, the CD baseline algorithm achieving success in herding a population of 15 was a fortunate event. This demonstrates the need for further testing of the CD Baseline algorithm under a variety of conditions. Conversely, the FAT baseline algorithm demonstrates a consistent trajectory across different population sizes, with an initial spike in radius followed by a reduction and relatively stable behavior over time. This pattern is a result of the UAV initially flying over the flock to approach the furthest animal, effectively dividing the flock into smaller groups. Subsequently, the algorithm gradually reunites the flock over time.

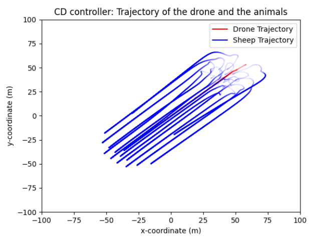
   Regarding the vision-based algorithms, the CD Vision algorithm exhibits varying radius patterns as it switches between modes for a population size of 5. However,



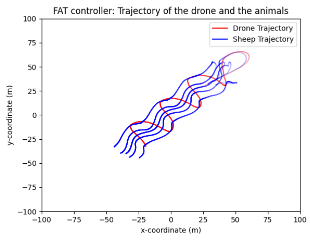
(a) CD Baseline on 5 animals



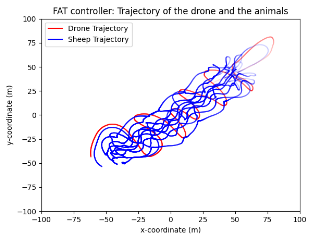
(b) CD Baseline on 10 animals



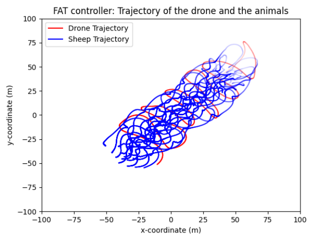
(c) CD Baseline on 15 animals



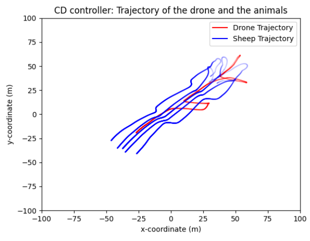
(d) FAT Baseline on 5 animals



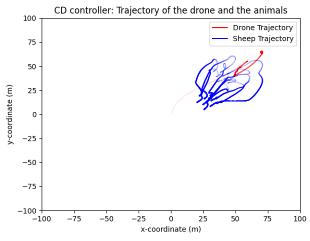
(e) FAT Baseline on 10 animals



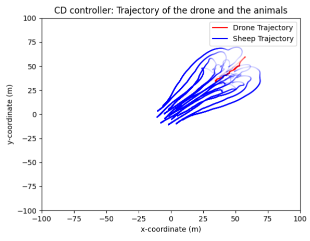
(f) FAT Baseline on 15 animals



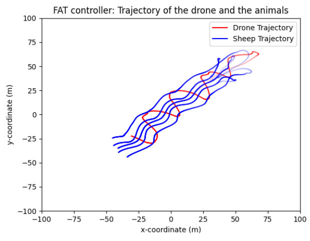
(g) CD Vision on 5 animals



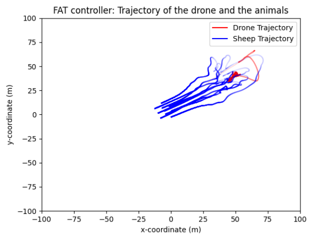
(h) CD Vision on 10 animals



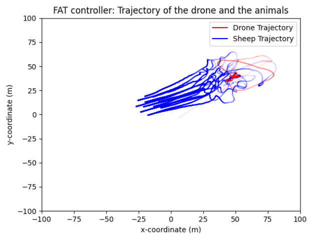
(i) CD Vision on 15 animals



(j) FAT Vision on 5 animals



(k) FAT Vision on 10 animals

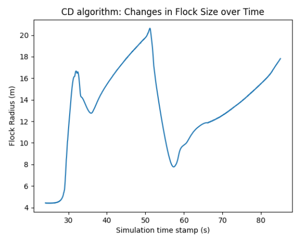


(l) FAT Vision on 15 animals

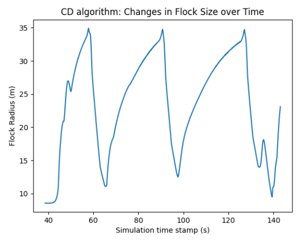
Figure 12: Herding Trajectory of the algorithms in scenario 1

this pattern deviates when herding fails for higher population sizes. Similarly, the FAT Vision algorithm follows a comparable trend. Notably, even when herding is successful, the CD Vision algorithm does not conform to the same flock radius pattern observed in its baseline algorithm. On the other hand, the FAT Vision algorithm demonstrates a slightly similar pattern.

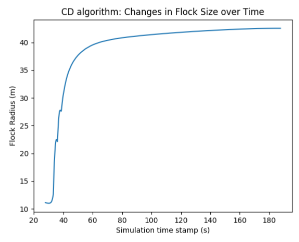
6. **Number of Detection per frame:** The number of animals detected by the camera in each frame over the duration of herding by the vision algorithm is



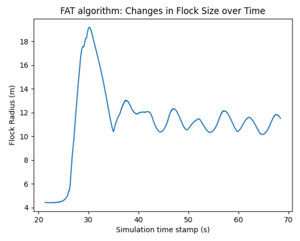
(a) CD Baseline with 5 animals



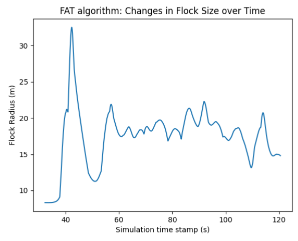
(b) CD Baseline with 10 animals



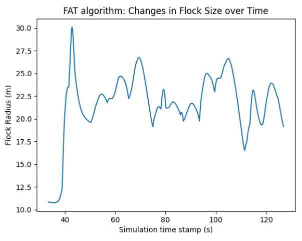
(c) CD Baseline with 15 animals



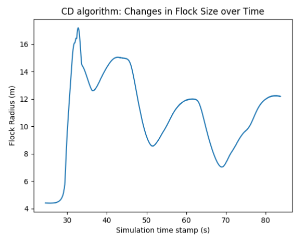
(d) FAT Baseline with 5 animals



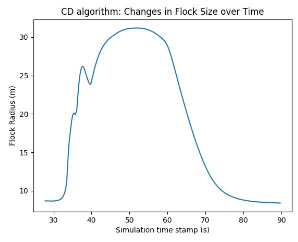
(e) FAT Baseline with 10 animals



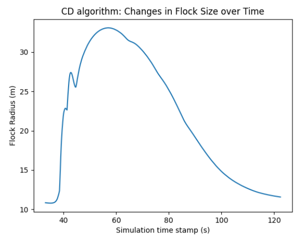
(f) FAT Baseline with 15 animals



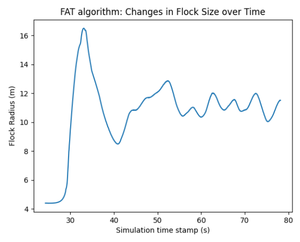
(g) CD Vision with 5 animals



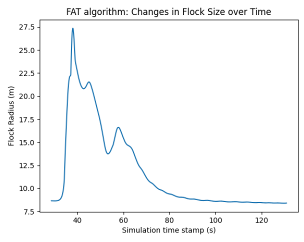
(h) CD Vision with 10 animals



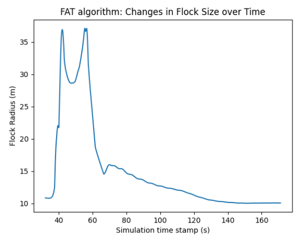
(i) CD Vision with 15 animals



(j) FAT Vision with 5 animals



(k) FAT Vision with 10 animals

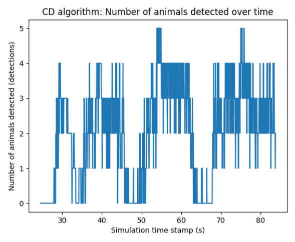


(l) FAT Vision with 15 animals

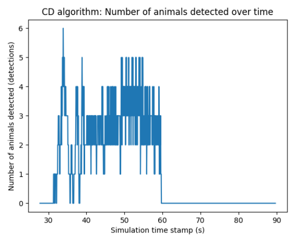
Figure 13: Flocking radius plots for the algorithms in scenario 1

shown in Figure 14. These graphs correspond to the simulation attempts of the vision algorithms for which the herding trajectories and the flocking radius graphs are illustrated above.

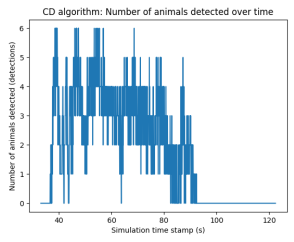
By establishing a correlation between the provided data, herding trajectories, and flock radii, it becomes evident that the vision algorithms struggle to achieve successful herding outcomes in scenarios involving higher animal population sizes. This difficulty arises due to the algorithms encountering challenges in



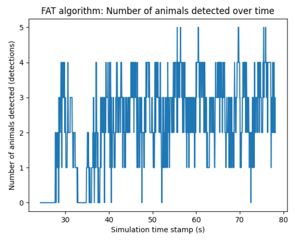
(a) CD Vision with 5 animals



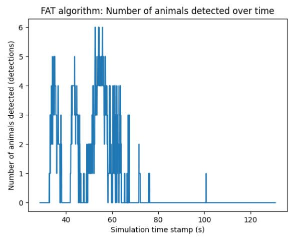
(b) CD Vision with 10 animals



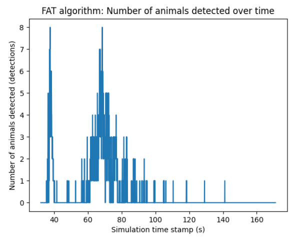
(c) CD Vision with 15 animals



(d) FAT Vision with 5 animals



(e) FAT Vision with 10 animals



(f) FAT Vision with 15 animals

Figure 14: Number of detections per frame for the vision algorithms in scenario 1

tracking and detecting animals within the simulation. Specifically, the UAV can become stuck in a position where the animals are not currently present primarily because the last known position significantly deviates from the actual animal locations.

Several factors contribute to this discrepancy. Firstly, the detection algorithm may fail to successfully identify certain animals, leading to gaps in tracking. Additionally, if the animals remain outside the camera's field of view for an extended period, their positions may become misaligned with the last known information. Consequently, the UAV relies on outdated data and returns to the previous known position in an attempt to herd animals that are no longer present in that location. As a result, the UAV gradually moves away from the herd, eventually losing track of them entirely.

This unfortunate outcome manifests in two potential scenarios. In one scenario, when other animals besides the target animal remain within the camera's field of view, the UAV may hover indefinitely, unable to discern the precise location of the target animal. In the other scenario, if no animals are detected for a continuous duration of 30 seconds, the simulation terminates.

These challenges highlight the impact of inaccurate animal tracking and detection on the effectiveness of the vision algorithms, especially in scenarios with higher animal population sizes.

In conclusion, for scenario 1, the FAT Baseline and FAT Vision algorithms exhibit faster performance but cover longer distances, indicating excessive movements compared to the CD algorithms. The herding trajectories also validate this observation.

Moreover, the FAT Vision algorithm outperforms the CD Vision algorithm in general and even the CD Baseline algorithm in terms of herding time. However, further data from other scenarios is needed to fully evaluate the potential of the FAT Vision algorithm.

4.2.2 Scenario 2: Loosely Packed, Slow Animals

This scenario represents stubborn animals that are harder to herd and repel due to their low velocity and weak escape rule strength. The flocking parameters for this scenario are as follows:

$m_c = 1.25$, $m_s = 5.5$, $m_a = 0.0$, $m_e = 1.5$, $m_{cp} = 3.5$, $m_{sp} = 16.5$, $m_{ap} = 0.125$, $v_{max} = 2.5$, $v_{min} = -2.5$, $r_a = 6.1833$, $r_{collect_5} = 20.68$, $r_{collect_{10}} = 34.77$, $r_{collect_{15}} = 47.13$, $r_{drive_5} = 10.57$, $r_{drive_{10}} = 13.32$, $r_{drive_{15}} = 15.25$

The above parameters result in a flock that behaves in a way illustrated by Figure 15. The reason is that the separation forces are higher than the cohesive forces compared to scenario 1. The escape rule coefficient is also lower, which results in the animals not being repelled effectively by the predator. The multiplier rule coefficients further increase the separation amongst the animals in the presence of a predator and alignment rule coefficient is much weaker here. Clipping the velocity between $v_{max} = 2.5$ and $v_{min} = -2.5$ prevents the animals from running faster. This results in an increased difficulty in herding the animals. We can also notice that the agent-to-agent interaction distance r_a is higher in this scenario. The gap between the collect and drive radius thresholds is slightly higher than scenario 1. Thus, the animals are loosely packed, harder to herd and slightly harder to keep together compared to scenario 1.

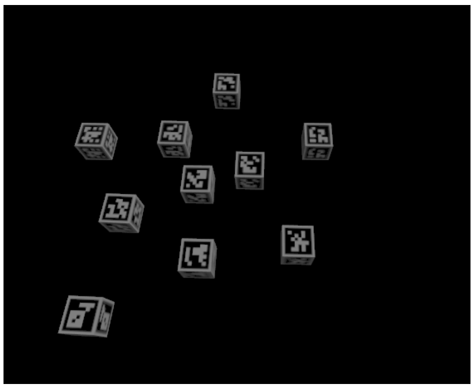


Figure 15: Scenario 2: Loosely Packed Animals

The following are the observations made while conducting the simulation with the above parameters. :

1. **Herding Success Rate:** The herding success rate for scenario 2, observed over 10 repetitions for each herding algorithm on different population sizes, is presented in Table 4.

	Animal Population		
	5 Animals	**10 Animals**	**15 Animals**
CD Baseline	100%	0%	0%
FAT Baseline	100%	100%	70%
CD Vision	10%	0%	0%
FAT Vision	100%	90%	20%

Table 4: Herding Performance for scenario 2

From this data, it is evident that the CD Baseline algorithm demonstrated limited success in herding, achieving 100% herding success only for the scenario with 5 animals. It faced challenges in achieving successful herding as the animal population increased. The FAT Baseline algorithm has showcased higher herding success rates. However, the success rate decreased to 70% for 15 animals, indicating increased difficulty in herding a larger population. The CD Vision algorithm exhibits lower herding success rates when compared to its baseline equivalent algorithm. One of the most surprising observations is how the FAT vision algorithm is able to achieve a high success rate even in the scenario involving 10 animals, which is better than its performance in the previous scenario. This suggests that although the animals are more stubborn in this scenario, the FAT Vision algorithm is able to herd them more effectively. However, the performance drops once the population increases to 15. This data alone is not enough to conclude the reason for the failure.

2. **Average Herding time:** The average time taken by the herding algorithms to complete a successful simulation is shown in Table 5. The tests are conducted in the same way tests were conducted in scenario 1 by collecting the average of 10 herding attempts per situation.

	Animal Population		
	5 Animals	**10 Animals**	**15 Animals**
CD Baseline	125.0	-	-
FAT Baseline	107.77	230.21	279.75
CD Vision	146.90	-	-
FAT Vision	119.95	247.74	299.32

Table 5: Average herding time in seconds for scenario 2

From this table, we can infer that the CD Baseline algorithm demonstrates a relatively shorter average herding time for the scenario with 5 animals compared to the CD Vision Algorithm, but it is slower than the FAT Algorithm variants. The FAT Vision algorithm is slower than the FAT baseline algorithm. Both the FAT algorithms showcase increasing herding times with an increase in the population sizes. The CD Vision algorithm is the slowest algorithm out of all and manages to herd only a small group of 5 animals.

Overall, we can observe that the time taken by the algorithms to herd the flock in this scenario is at least twice the time taken by the algorithms to herd the animals in scenario 1 and can be as much as thrice the time taken by the same algorithm in a similar situation in scenario 1. This is indicative of the stubborn behavior of the animals, requiring more effort to herd them towards the goal position.

Yet another observation is that the herding time for the FAT algorithms in the case of 15 animals is very close to the 300-second mark at which the herding is considered to be a failure. Thus, the herding success percentage could have been higher for the FAT algorithms if they were given more time to complete the herding process.

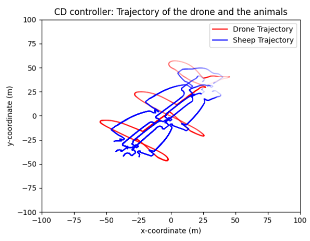
3. **Average Distance Travelled:** The average distance travelled by the UAV while performing the herding process for the 2_{nd} scenario is presented in Table 6.

	Animal Population		
	5 Animals	**10 Animals**	**15 Animals**
CD Baseline	499.11	-	-
FAT Baseline	685.62	1469.71	1770.46
CD Vision	517.78	-	-
FAT Vision	636.45	1171.84	1352.66

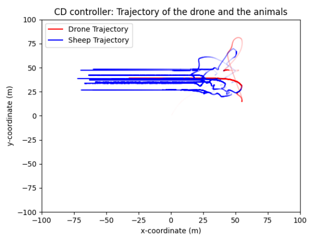
Table 6: Average distance travelled in meters for scenario 2

Similar to the previous scenario, the CD Baseline algorithm travels a shorter distance than the FAT Baseline algorithm when it succeeds in herding the animals despite taking a longer time as seen above. This is indicative of slower movement in the CD baseline algorithm. It is interesting to observe that the FAT vision algorithm travels a shorter distance than the FAT Baseline algorithm. However, the explanation for this observation is not clear.

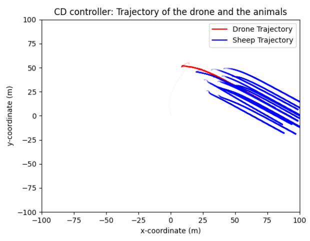
4. **Herding Trajectory:** The herding trajectories of the baseline algorithms and the vision based algorithms when tested on scenario 2 for different number of animals are shown in Figure 16.



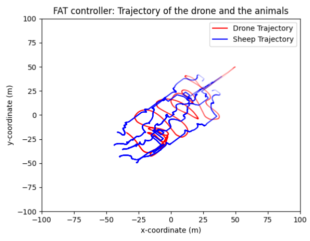
(a) CD Baseline on 5 animals



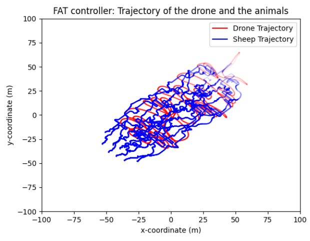
(b) CD Baseline on 10 animals



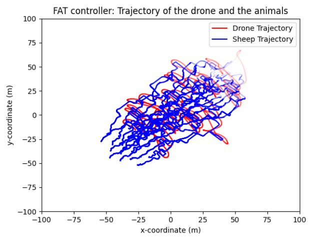
(c) CD Baseline on 15 animals



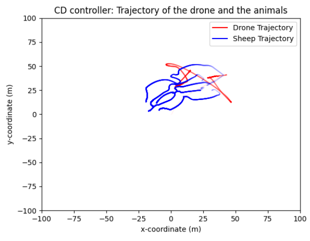
(d) FAT Baseline on 5 animals



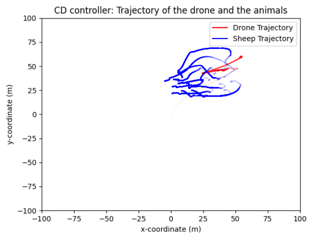
(e) FAT Baseline on 10 animals



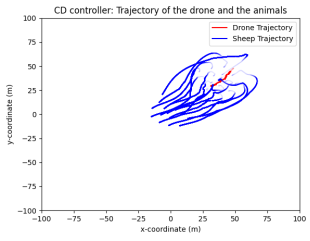
(f) FAT Baseline on 15 animals



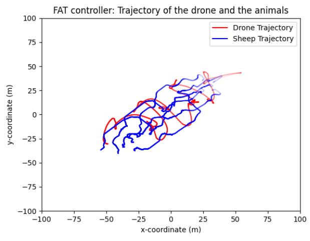
(g) CD Vision on 5 animals



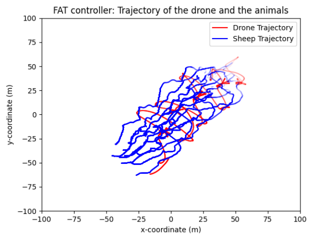
(h) CD Vision on 10 animals



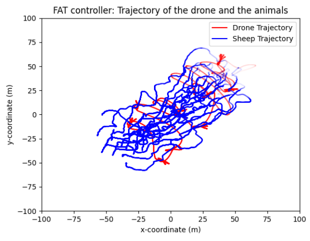
(i) CD Vision on 15 animals



(j) FAT Vision on 5 animals



(k) FAT Vision on 10 animals



(l) FAT Vision on 15 animals

Figure 16: Herding Trajectory of the algorithms in scenario 2

From these trajectories we can understand that the herding process is less straightforward than herding the animals in scenario 1. Both the FAT algorithms manage to herd the flock very close to the goal position irrespective of the herd size. The FAT vision algorithm does not lose track of the animals even when the flock size increases. This gives more evidence to the hypothesis that the success rates for these algorithms would be higher if they were given more time to herd the animals. However, due to the wide discrepancy that occurs between the actual position of the animals and their last known position, the motion of the UAV is very violent which can be seen through the erratic trajectories at the turns in the trajectory of the FAT vision algorithm. The CD algorithms however have difficulty in herding the flocks. The CD baseline algorithm struggles to herd the animals in the direction of the goal position for higher population sizes. This indicated that the UAV never reached the drive position as all the time is spent on trying to collect the animals and bring the flock radius below the threshold r_{drive_N}. The CD vision algorithm loses track of the animals even for lower population sizes.

5. **Flock Radius:** The change in the actual radius of the flock over time when the baseline and the vision based algorithms are deployed in scenario 2 for different animal population sizes are shown in Figure 17. These graphs correspond to the herding attempts in the herding trajectory graphs shown above.

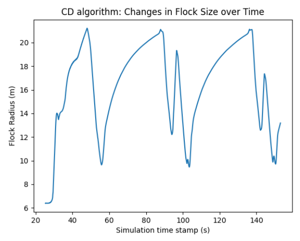
   In this analysis, the CD baseline algorithm exhibits the collect and drive motion observed in scenario 1 for a population size of 5, as indicated by the increasing and decreasing radius of the flock in different modes. However, this behavior is not reflected in higher population sizes. This is exactly similar to the observation made in scenario 1. This is because of the effect of the second multiplier $p(x)$, which amplifies m_{sp} when the UAV approaches them. Thus, the animals tend to run away from each other in the presence of a UAV, making it much harder to collect them. The UAV starts in collect mode, but it is not able to collect the animals together such that the flock radii drop below r_{drive_N} in order to switch to the drive mode. This confirms the hypothesis made above from the herding trajectories and also proves that the successful herding in scenario 1 for a population size of 15 was a fortunate event.

   Thus, it is harder to collect loosely packed animals, which have stronger separative forces and weaker cohesive forces acting between them. Higher population sizes can further aggravate this problem, even for tightly packed animals, as observed in scenario 1.

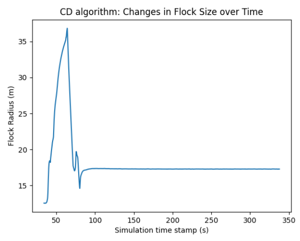
   The CD vision algorithm exhibits the collect and drive behavior initially for the situation with 5 animals in scenario 2, but it eventually loses track of the animals, and the flock radius decreases. Similarly, in higher population sizes, once the UAV loses track of the animals, they start to collect together again since the predator is absent and the effect of $p(x)$ is minimal for m_{sp} to have any pronounced effect. This can also be observed in the herding trajectory curves for the CD vision simulations. This proves that the collect and drive strategy is ineffective for loosely packed animals, which spread out further in the presence of a predator.

   In comparison, the FAT baseline algorithm shows a similar trend across all population sizes, where the radii of the flocks increase initially but decrease towards the end of the simulation as all the animals approach the goal position. This effect is also loosely observed in the FAT vision algorithm.

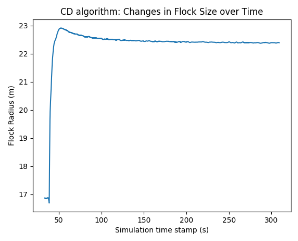
6. **Number of Detection per frame:** The number of animals detected by the



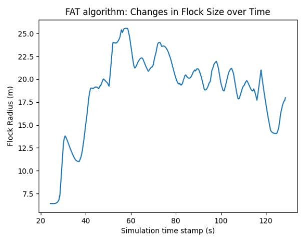
(a) CD Baseline with 5 animals



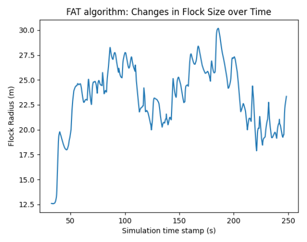
(b) CD Baseline with 10 animals



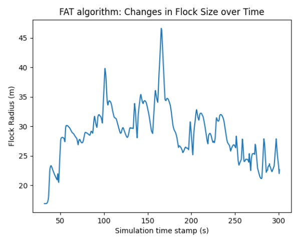
(c) CD Baseline with 15 animals



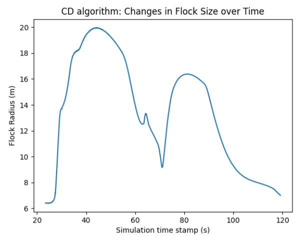
(d) FAT Baseline with 5 animals



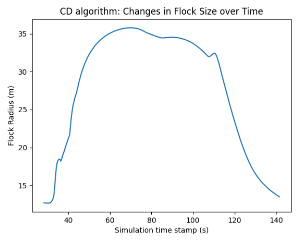
(e) FAT Baseline with 10 animals



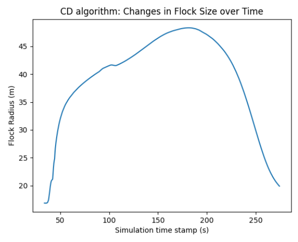
(f) FAT Baseline with 15 animals



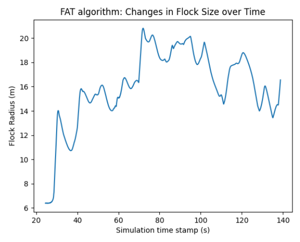
(g) CD Vision with 5 animals



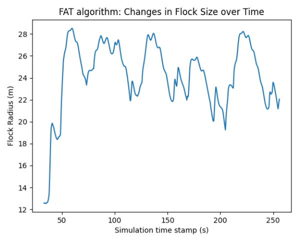
(h) CD Vision with 10 animals



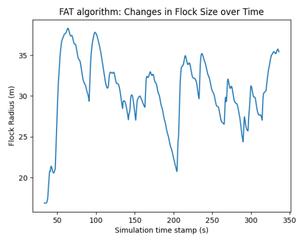
(i) CD Vision with 15 animals



(j) FAT Vision with 5 animals



(k) FAT Vision with 10 animals

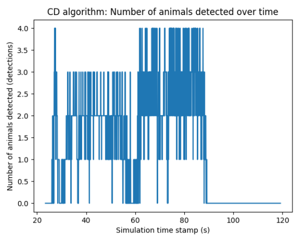


(l) FAT Vision with 15 animals

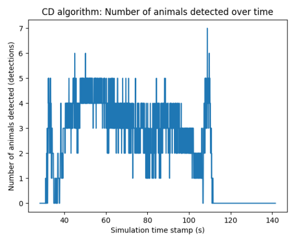
Figure 17: Flocking radius plots for the algorithms in scenario 2

camera in each frame over the duration of herding by the vision algorithm is shown in Figure 18. These graphs correspond to the simulation attempts of the vision algorithms for which the herding trajectories and the flocking radius graphs are illustrated in this scenario.

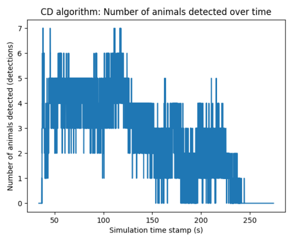
Correlating the detection data with the herding trajectories and the flock radii curves illustrate that the CD vision algorithm loses track of animals for loosely packed animals much easily when compared to tightly packed animals. However,



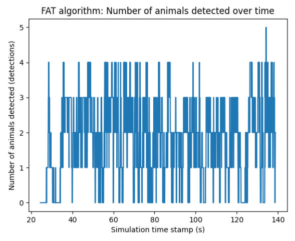
(a) CD Vision with 5 animals



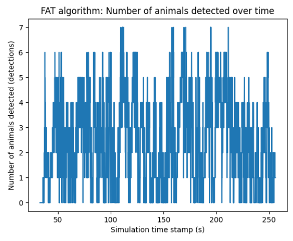
(b) CD Vision with 10 animals



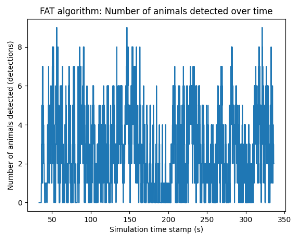
(c) CD Vision with 15 animals



(d) FAT Vision with 5 animals



(e) FAT Vision with 10 animals



(f) FAT Vision with 15 animals

Figure 18: Number of detections per frame for the vision algorithms in scenario 2

it is surprising to note that the FAT Vision algorithm manages to successfully herd the loosely packed animals more successfully than the closely packed animals. While Intuitively it seems easier to keep track of closely packed animals, it was observed during the simulation that closely packed animals increased the occlusion of some animals, causing the failure of the object detection algorithm. This resulted in misalignment between the last known position of the animals and the actual position of the animals. However, a loosely packed flock makes has lesser occlusion, resulting in successful detection of the animals as shown in the plot of detections per frame. Despite the considerably better performance shown by the FAT vision algorithm in scenario 2, we can still see some dips in the number of animals detected by the camera in certain points in the simulation which correspond to increasing deviation between the real positions and the last known positions of the animals. It is also important to note that at any point in time, the detected animals are usually lower than the actual animals present in the simulation, indicating misalignment between the actual and perceived positions even further.

We can conclude from these scenarios we have studied so far, that while the FAT algorithms may traverse a longer distance due to excessive movements, they are capable of successful herding in a wide variety of scenarios as opposed to the CD algorithms which are less effective in scenarios involving loosely packed animals. We can also conclude that the FAT algorithms could have achieved a higher success rates when allowed to perform herding for a longer duration. An alternative approach would involve speeding up herding time by increasing the value of the escape rule coefficient m_e which would result in the animals running away faster. In the real world, this would translate to employing various strategies such as playing the audio of predatory animals to induce the animals to be more scared.

4.2.3 Scenario 3: Widely spread, Fast Animals

This scenario represents animals that are much loosely packed compared to the other scenarios, making it very hard to keep track of all the animals within the field of view of the camera as they are spread out very far from each other. The flocking parameters for this scenario are as follows:

$m_c = 0.5, \quad m_s = 6.0, \quad m_a = 0.0, \quad m_e = 3.0, \quad m_{cp} = 2.5, \quad m_{sp} = 14.5, \quad m_{ap} = 2.0, \quad v_{max} = 50.0, \quad v_{min} = -50.0, \quad r_a = 8.9, \quad r_{collect_5} = 29.76, \quad r_{collect_{10}} = 50.05, \quad r_{collect_{15}} = 67.84, \quad r_{drive_5} = 15.22, \quad r_{drive_{10}} = 19.17, \quad r_{drive_{15}} = 21.95$

The above parameters result in a flock that behaves in a way illustrated by Figure 19. In this scenario the cohesive force between the animals is extremely weak when compared to the forces of separation due to the values of m_c and m_s. This is further exacerbated in the presence of a predator due to the huge gap between m_{cp} and m_{sp}. The value of r_a is higher than its values observed in the previous scenarios. The gap between the collect and drive threshold determined by the r_{drive} and $r_{collect}$ values makes it more harder for the UAV to switch from collect mode to drive mode as the UAV has to spend a lot more time to decrease the radius of the flock which is harder when the forces of separation are high.

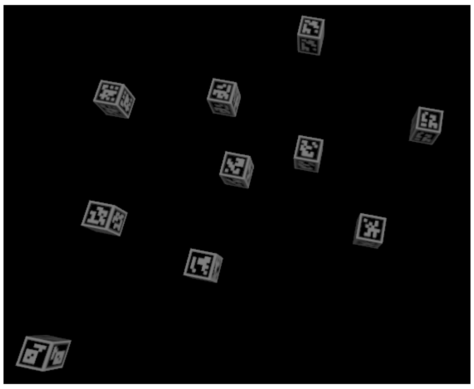


Figure 19: Scenario 3: Widely spread Animals

The following are the observations made while conducting the simulation with the above parameters:

1. **Herding Success Rate:** The herding success rate for scenario 3, observed over 10 repetitions for each herding algorithm on different population sizes, is presented in Table 7.

	Animal Population		
	5 Animals	**10 Animals**	**15 Animals**
CD Baseline	100%	0%	0%
FAT Baseline	90%	90%	60%
CD Vision	0%	0%	0%
FAT Vision	0%	0%	0%

Table 7: Herding Performance for scenario 3

From this data, we can observe that while the CD Baseline algorithm demonstrated successful herding outcomes for the scenario with 5 animals, it encountered challenges in herding larger animal populations. The FAT Baseline algorithm achieved relatively high herding success rates for 5 and 10 animals, but faced greater difficulty in herding 15 animals. However, the CD Vision algorithm and the FAT Vision algorithm were unable to achieve herding success for any of the population sizes. This suggests that herding a widely spread flock in general is a much harder problem in general as even having the formation of the exact position of the animals doesn't guarantee successful herding.

2. **Average Herding time:** The average time taken by the herding algorithms to complete a successful simulation is shown in Table 8. The tests are conducted in the same way tests were conducted in scenario 1 and 2 by collecting the average of 10 herding attempts per situation.

	Animal Population		
	5 Animals	**10 Animals**	**15 Animals**
CD Baseline	200.91	-	-
FAT Baseline	130.31	209.01	264.90
CD Vision	-	-	-
FAT Vision	-	-	-

Table 8: Average herding time in seconds for scenario 3

From this table, we can infer that the CD baseline algorithm and the FAT baseline algorithms follow a similar trend noticed in scenario 2. However, the CD algorithms takes a much longer duration in this scenario. The herding time of the FAT algorithm is comparable to those of scenario 2. No data is available to analyse the vision algorithms as they fail to perform herding in this scenario.

3. **Average Distance Travelled:** The average distance travelled by the UAV while performing the herding process for the 3_{rd} scenario is presented in Table 9.

	Animal Population		
	5 Animals	10 Animals	15 Animals
CD Baseline	1672.26	-	-
FAT Baseline	1043.69	1666.44	2133.97
CD Vision	-	-	-
FAT Vision	-	-	-

Table 9: Average distance travelled in meters for scenario 3

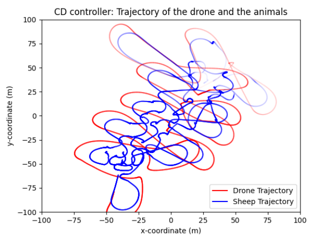
Unlike to the previous scenario, the CD Baseline algorithm travels a longer distance than the FAT Baseline algorithm when it succeeds in herding the animals. For the FAT baseline algorithm, we can see that the herding time increases with the population size of the animals which is consistent with our findings from the previous scenarios.

4. **Herding Trajectory:** The herding trajectories of the baseline algorithms and the vision based algorithms when tested on scenario 3 for different number of animals are shown in Figure 20.

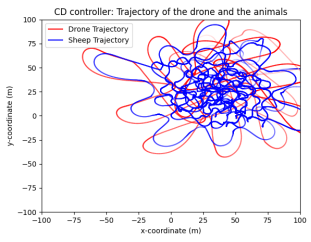
   These trajectories illustrate the difficulty experienced by the CD algorithm to keep the flock together. A lot of time is spent trying to bring the animals together instead of herding them. The shortcomings of the CD baseline algorithm are overcome by the FAT baseline algorithm. The FAT baseline algorithm tries to herd the animals towards the goal instead of bringing them together. This makes the FAT baseline algorithm more effective. In the case of the Vision based algorithms, we can note that the animals are widely spread out that it becomes impossible for the camera to keep track of all the animals. While the FAT vision algorithm performed well in a less tightly packed scenario, it struggles to keep track of the animals when they are widely spread as they move beyond the field of view of the UAV's camera. The CD vision algorithm is ineffective in this scenario which is consistent with our hypothesis that they are no effective for herding loosely packed ground of animals. This is also consistent with the findings of Tsunoda et al. [22].

5. **Flock Radius:** The change in the actual radius of the flock over time when the baseline and the vision based algorithms are deployed in scenario 3 for different animal population sizes are shown in Figure 21. These graphs correspond to the herding attempts in the herding trajectory graphs shown above.

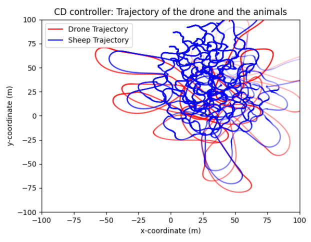
   In the above graphs, we can see the flock radii of the flocks being herded by the CD baseline algorithm for different population sizes. For the population size of 5, it switches between collect and drive modes regularly and eventually completes the herding successfully. For the higher population sizes, while it might look like the UAV is rapidly switching between collect and drive modes, the UAV switches to drive mode only once during the herding of 10 animals in the beginning when the flock radius dips below 20 meters. It immediately



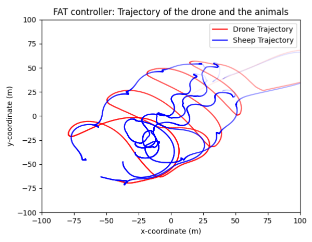
(a) CD Baseline on 5 animals



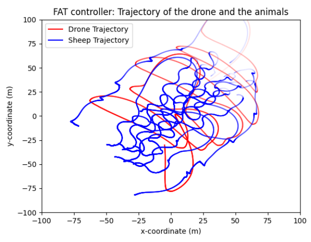
(b) CD Baseline on 10 animals



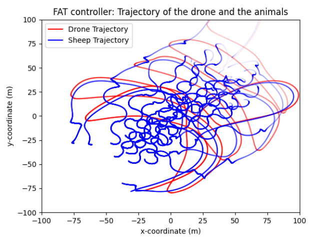
(c) CD Baseline on 15 animals



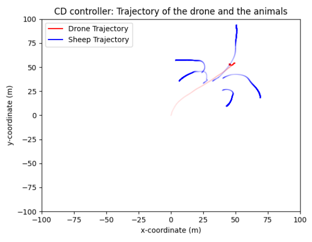
(d) FAT Baseline on 5 animals



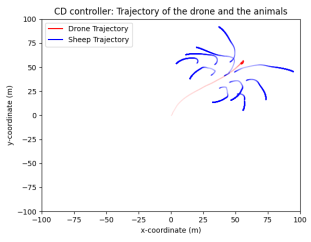
(e) FAT Baseline on 10 animals



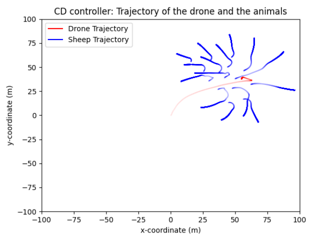
(f) FAT Baseline on 15 animals



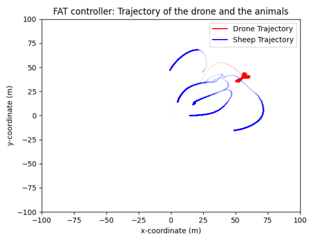
(g) CD Vision on 5 animals



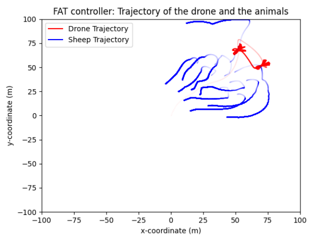
(h) CD Vision on 10 animals



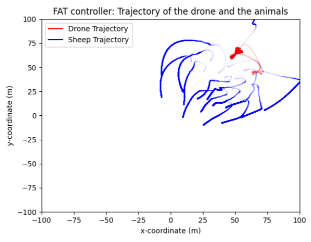
(i) CD Vision on 15 animals



(j) FAT Vision on 5 animals



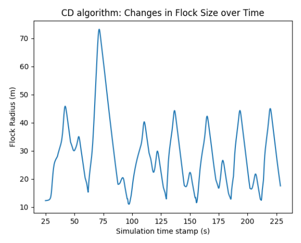
(k) FAT Vision on 10 animals



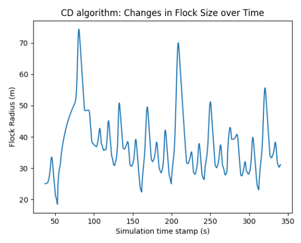
(l) FAT Vision on 15 animals

Figure 20: Herding Trajectory of the algorithms in scenario 3

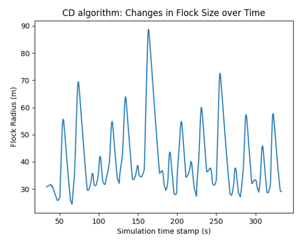
switches to collect mode in a few seconds when the radius increases beyond 50 meters and it never switches back to drive mode after that. The UAV simply spends its time keeping the animals together as shown in the herding trajectory. For the 15 animal scenario, it doesn't switch to drive mode even once as the flock radius needs to dip below 22 meters, which isn't the case. Thus, the CD algorithm doesn't work for loosely packed scenarios and this is worsened by higher population sizes. The findings are consistent with the previous scenarios. The CD vision algorithm gets stuck in the center of the mass of the flock,



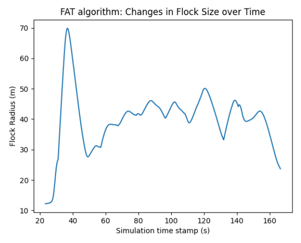
(a) CD Baseline with 5 animals



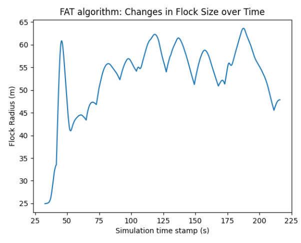
(b) CD Baseline with 10 animals



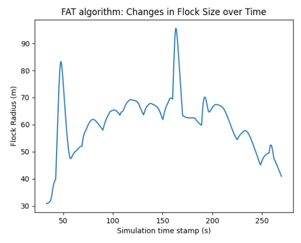
(c) CD Baseline with 15 animals



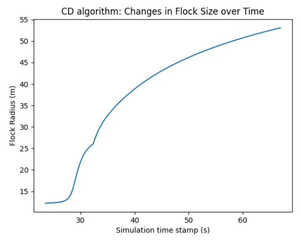
(d) FAT Baseline with 5 animals



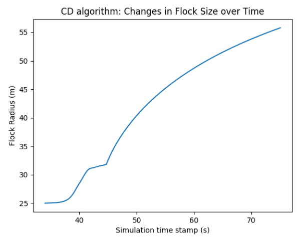
(e) FAT Baseline with 10 animals



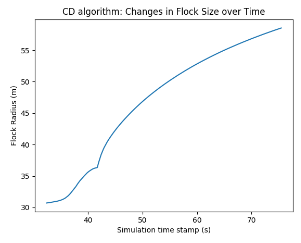
(f) FAT Baseline with 15 animals



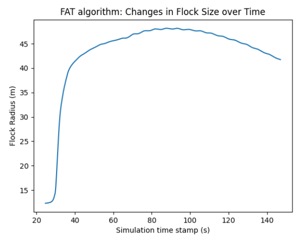
(g) CD Vision with 5 animals



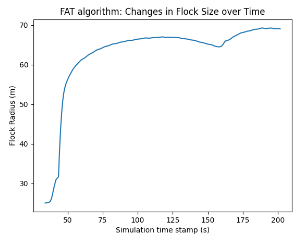
(h) CD Vision with 10 animals



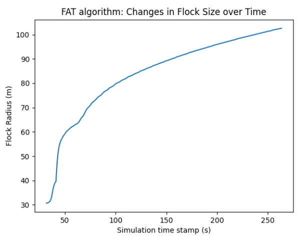
(i) CD Vision with 15 animals



(j) FAT Vision with 5 animals



(k) FAT Vision with 10 animals

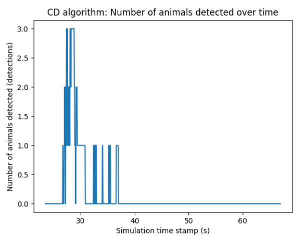


(l) FAT Vision with 15 animals

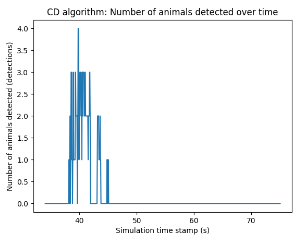
Figure 21: Flocking radius plots for the algorithms in scenario 3

preventing the animals from grouping together and reaches and equilibrium. Animals are completely repelled from its field of view and the herding terminates due to losing track of all the animals. The FAT baseline algorithm was the only algorithm capable of achieving successful herding across all population sizes, making it the most effective and reliable herding solution out of all the algorithms being studied. It can be seen that the movement of the UAV is more erratic for higher population sizes due to the difficulty of herding a widely spread flock. The FAT vision algorithm simply loses track of the animals like the CD vision algorithm as the animals are not within the field of view of the UAV's camera.

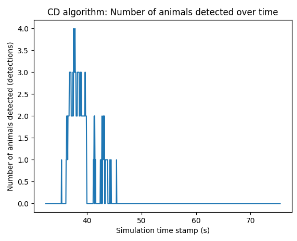
6. **Number of Detection per frame:** The number of animals detected by the camera in each frame over the duration of herding by the vision algorithm is shown in Figure 22. These graphs correspond to the simulation attempts of the vision algorithms for which the herding trajectories and the flocking radius graphs are illustrated above.



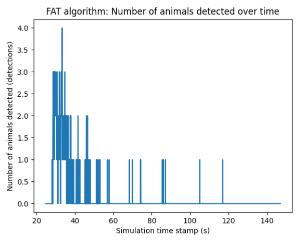
(a) CD Vision with 5 animals



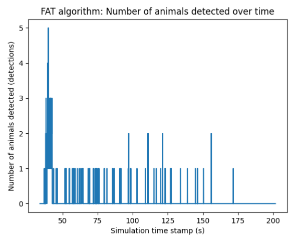
(b) CD Vision with 10 animals



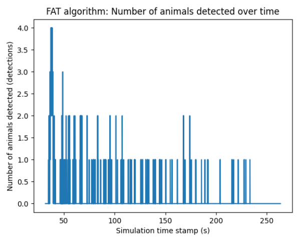
(c) CD Vision with 15 animals



(d) FAT Vision with 5 animals



(e) FAT Vision with 10 animals



(f) FAT Vision with 15 animals

Figure 22: Number of detections per frame for the vision algorithms in scenario 3

From the detections data, we can infer that the CD vision algorithm immediately loses track of the animals, while the FAT vision algorithm keeps track of a few animals before it loses track of all the animals as well.

From this scenario, we can conclude that CD algorithms are not effective in herding widely spread animals and the FAT algorithm is effective only when the actual positions of the animals are already known. Since this is impossible to replicate in the real world with a single UAV, we need to find a better approach to herding a large number of loosely packed animals using a single UAV using camera based localisation.

5 Discussion and Future Work

5.1 Inferences

The performance metrics observed upon deploying the proposed algorithms, as well as the answers to the research questions set forth at the beginning of this study, are discussed below.

- **Success Rate:** From the herding success rates provided for the three scenarios and varying animal populations, it is observed that the performance of the single-UAV herding strategies (CD Vision and FAT Vision) generally decreases as the animal population increases, in contrast to the more consistent success rates of the baseline two-UAV strategies (CD Baseline and FAT Baseline). For instance, in Scenario 1, both CD Vision and FAT Vision exhibit declines in success rate with an increasing animal population, dropping to 0% for 15 animals, while the two-UAV strategies maintain a 100% success rate across all population sizes. This pattern is similar in Scenarios 2 and 3, with the single-UAV strategies struggling particularly in Scenario 3, where they achieved 0% success for all population sizes. Hence, in terms of herding success rate, the baseline two-UAV strategies generally outperform the proposed single-UAV strategies, especially as the animal population increases. However, the FAT Vision algorithm shows promise for loosely packed flocks, where it achieved a higher success rate than both the baseline and vision-based CD algorithms on average.

- **Herding Time:** In Scenario 1, CD Vision took approximately 1.71% longer than CD Baseline to herd 5 animals, while FAT Vision took approximately 15.72% and 7.08% longer than FAT Baseline to herd 5 and 10 animals, respectively. In Scenario 2, CD Vision took approximately 17.52% longer than CD Baseline for 5 animals. FAT Vision took about 11.30%, 7.61%, and 6.99% longer than FAT Baseline to herd 5, 10, and 15 animals, respectively. Based on these percentage differences, we can conclude that the proposed single-UAV herding strategies take slightly longer to herd the animals compared to the baseline strategies in most cases. However, the extent of this difference varies. Therefore, we can say that the time efficiency of the proposed single-UAV herding strategies is generally lower than that of the baseline strategies, as they take longer to successfully herd the animals across different scenarios and population sizes.

- **Distance Travelled:** In Scenario 1, CD Vision travelled approximately 8.96% more than CD Baseline for 5 animals, while FAT Vision travelled approximately 6.68% more for 5 animals and 11.17% less for 10 animals compared to FAT Baseline. In Scenario 2, CD Vision travelled approximately 3.74% more than CD Baseline for 5 animals. FAT Vision travelled approximately 7.17%, 20.27%, and 23.60% less than FAT Baseline for 5, 10, and 15 animals, respectively. In Scenario 3, both CD Vision and FAT Vision failed to herd any of the animals, hence no comparison can be made with the baseline strategies. Therefore, we can conclude that the proposed single-UAV herding strategies generally travel either a comparable or a greater distance than the baseline strategies when successful. However, the FAT Vision strategy shows a pattern of travelling shorter distances than the FAT Baseline strategy across different scenarios and population sizes. More data is required to strengthen this claim.

- **Herding Capacity:** In Scenario 1, the CD Vision strategy successfully herded 5 animals but failed with larger populations, indicating its herding capacity in this scenario is limited to 5 animals. The FAT Vision strategy successfully herded 5 and 10 animals but failed with 15 animals, suggesting its herding capacity in this scenario is 10 animals. In Scenario 2, the CD Vision strategy was only successful with 5 animals, showing its herding capacity to be 5 animals. The FAT Vision strategy successfully herded 5, 10, and 15 animals, indicating that its herding capacity in this scenario is 15 animals. In Scenario 3, both CD Vision and FAT Vision strategies failed to herd any of the animals, suggesting a herding capacity of 0 animals for this scenario. In conclusion, the herding capacity of the proposed single-UAV herding strategies, CD Vision and FAT Vision, varies depending on the scenario and the specific behaviors of the animals. The FAT Vision strategy generally exhibits a higher herding capacity compared to the CD Vision strategy. However, in more challenging scenarios, both strategies struggle to herd the animals effectively. In Scenario 3, both CD Vision and FAT Vision failed to herd any of the animals, hence no comparison can be made with the baseline strategies.

In conclusion, while the proposed single-UAV strategies show some promise, the baseline two-UAV strategies generally outperform them across all considered metrics, especially as the animal population increases. However, the FAT Vision strategy does display some advantages, such as a higher herding capacity and shorter travel distances under certain conditions, suggesting potential areas for further exploration and improvement. This answers our primary research question.

5.2 Observations and Limitations

In this study, we observed that the Furthest Agent Targeting (FAT) algorithms generally perform better than their Collect and Drive algorithm counterparts. This is also consistent with the observations made by Tsunoda et al. [22] in their study. One of the main reasons why the FAT algorithm performs better in general is that, unlike the Collect and Drive algorithm, the UAV experiences a repulsive force from the furthest animal if it gets close. This prevents the flock from breaking frequently, and the FAT algorithm doesn't waste its effort trying to constantly bring the animals together before proceeding to move them towards the goal. Instead, it directly steers all the animals towards the goal position. The Collect and Drive algorithm, however, switches between positions when it changes from the collect to drive mode. During this switch, the UAV often flies over the flock to get behind it, breaking the flock in the process. Additionally, the FAT algorithm accepts velocity setpoint commands instead of position setpoint commands. The velocity setpoint commands change gradually at every instant of the herding process instead of jumping abruptly. However, the Collect and Drive algorithm switches between two modes abruptly, and its position setpoints are spaced far from each other, resulting in an unstable flight path.

While the vision algorithms offer promise, there are some serious limitations that need to be addressed before deploying these methods into the real world. The first issue is that in order to keep track of the animals, we associate each animal with a unique ArUco marker to store their last known positions. Translating this into the real world can prove to be a challenging task since real-world object detection algorithms with object tracking are not capable of recognizing an animal, which was not spotted for an extended period of time, as the same animal which they detected a while ago before the animal went outside its field of view. Thus, we need to overcome this problem.

Another limitation of the proposed vision algorithm is the erratic trajectory of the UAV caused by a sudden change in the last known position of some animals once they are detected after extended periods of time. This abrupt change causes the velocity/position setpoint of the herding algorithm to change abruptly, translating to an erratic flight trajectory. Techniques to achieve smoother trajectories need to be pursued.

5.3 Future Work

Future work to improve this algorithm could involve incorporating reinforcement learning strategies into the herding algorithm, such as the proposed works of Nguyen et al. [14], [15]. An actor neural network, which accepts the last known position of the animals as input to predict the position and velocity setpoints for the UAV, could be used to train the reinforcement learning algorithm. A critic neural network could be used to evaluate the actor's actions. Another area of research could involve studying the active control of the UAV camera's gimbal mount to search for and keep the animals within its field of view, instead of relying on the UAV's movement to match that of the animals. While this study aspired to herd all the animals in a single attempt by keeping track of their positions, there is value in trying to herd the animals by herding multiple subgroups instead of the entire flock. This approach would allow us to perform herding without the position update algorithm, enabling us to overcome the shortcoming of misalignment between the last known position and the actual position observed in these algorithms for larger and denser groups.

To overcome the need to use unique ArUco markers, which is difficult to replicate in the real world, we could use predictive modeling to estimate the likely position of an animal when it is not visible. This could be based on the animal's previous movements, the movements of the rest of the flock, or environmental factors. When the animal reappears, the algorithm could use this predicted position to help re-identify the animal. The predictive modeling could also avoid abrupt changes in the last known position of the animals once they are detected after a long time, as the last known position estimate would be somewhat closer to the actual position than the last known position that has not been updated for extended periods of time. We could also use a ramp for the velocity or position setpoints to gradually change the setpoints instead of changing them abruptly in an instant to offer a smoother trajectory for the UAV. We could also implement path planning to calculate a path from the collect position to the drive position around the flock instead of over the flock to improve the performance of the collect and drive algorithm.

6 Conclusion

This study introduces a vision-based position tracking algorithm that can be integrated with herding algorithms and camera-based localization techniques. Two baseline algorithms, the Collect and Drive (CD) algorithm and the Furthest Agent Targeting (FAT) algorithm, were modified to incorporate the proposed vision-based approach. These algorithms were evaluated in three herding scenarios with varying animal population sizes.

The comparison between the proposed single-UAV herding strategies (CD Vision and FAT Vision) and the baseline two-UAV strategies (CD Baseline and FAT Baseline) reveals interesting insights. While the CD Vision algorithm consistently performs worse, the FAT Vision algorithm occasionally outperforms even the CD Baseline algorithm. However, the FAT Baseline algorithm consistently demonstrates superior performance. The success rates of the single-UAV strategies decline as the animal population increases, while the baseline strategies maintain more consistent success rates. The single-UAV strategies also tend to take a slightly longer time to complete the herding process compared to the baseline strategies. Additionally, the distance traveled by the single-UAV strategies is generally comparable to or greater than the baseline strategies, except for the FAT Vision algorithm, which travels shorter distances in some scenarios. The herding capacity varies depending on the scenario, with the FAT Vision algorithm displaying a higher capacity in certain cases.

The vision-based FAT algorithm outperformed the vision-based CD algorithm, particularly in loosely packed herds. It exhibited improved object detection accuracy and position tracking efficiency due to fewer occlusions among neighboring animals. However, the vision-based FAT algorithm faced challenges with widely dispersed animal populations, as many animals fell outside the UAV's camera field of view.

Despite these limitations, the vision-based FAT algorithm showed comparable performance to the FAT baseline algorithm, which has access to real-time position data for all animals, in terms of herding completion time and the distance traveled by the UAV. Hence, the vision-based FAT algorithm holds great potential for effectively herding animal flocks in real-world scenarios using a single UAV equipped with an integrated camera sensor.

Overall, this study provides valuable insights into integrating vision-based algorithms with herding strategies and highlights the advantages and limitations of the proposed approach. Further research and improvements are necessary to address the identified limitations and enhance the overall performance of vision-based herding algorithms in practical applications.

References

[1] Damage to agricultural crops caused by wildlife has doubled, *Statistics Sweden*. Online article. Available https://www.scb.se/en/finding-statistics/statistics-by-subject-area/agriculture-forestry-and-fishery/agricultural-production/production-of-cereals-dried-pulses-and-oil-seeds/pong/statistical-news/production-of-cereals-dried-pulses-and-oil-seeds-2020/ (accessed on 1.6.2023)

[2] C. W. Reynolds (1987), "Flocks, Herds, and Schools: A Distributed Behavioral Model". *ACM SIGGRAPH Computer Graphics*. 21. 25-34. DOI: https://doi.org/10.1145/280811.281008

[3] J.-M. Lien, O. Bayazit, R. Sowell, S. Rodriguez, and N. Amato (2004), "Shepherding behaviours," *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 4, pp. 4159-4164. DOI:https://doi.org/10.1109/ROBOT.2004.1308924

[4] C. Delgado-Mata, J. Ibáñez-Martínez, S. Bee, R. Ruiz-Rodarte, and R. Aylett (2007), "On the Use of Virtual Animals with Artificial Fear in Virtual Environments," *New Generation Computing.*, 25. 145-169. DOI:https://doi.org/10.1007/s00354-007-0009-5

[5] M. Barksten and D. Rydberg (2013), "Extending Reynolds' flocking model to a simulation of sheep in the presence of a predator," Bachelor's thesis, *School of computer science and communication, KTH - Royal Institute of Technology*, Stockholm, Sweden.

[6] D. Strömbom, R. P. Mann, A. M. Wilson, S. Hailes, A. J. Morton, D. J. T. Sumpter, and A. J. King (2014), "Solving the shepherding problem: heuristics for herding autonomous, interacting agents," *Journal of the Royal Society Interface* 11(100), 20147019., DOI: https://doi.org/10.1098/rsif.2014.0719

[7] S. Gade, A. A. Paranjape, and S.-J. Chung (2015), "Herding a flock of birds approaching an airport using an unmanned aerial vehicle," *AIAA guidance, navigation, and control conference.*, DOI: https://doi.org/10.2514/6.2015-1540

[8] N. K. Long, K. Sammut, D. Sgarioto, M. Garratt and H. A. Abbass (2020), "A Comprehensive Review of Shepherding as a Bio-Inspired Swarm-Robotics Guidance Approach," in *IEEE Transactions on Emerging Topics in Computational Intelligence.*, vol. 4, no. 4, pp. 523-537, DOI: https://doi.org/10.1109/TETCI.2020.2992778

[9] X. Li, H. Huang, A. V. Savkin, and J. Zhang (2022)"Robotic herding of farm animals using a network of barking aerial drones," *Drones*, 6(2), 29., DOI: https://doi.org/10.3390/drones6020029

[10] A. J. King , S. J. Portugal, D. Strömbom, R. P. Mann, J. A. Carillo, D. Kalise, G. Croon, H. Barnett, P. Scerri, R. Groß, D. P. Chadwick, and M. Papadopoulou (2023)"Biologically inspired herding of animal groups by robots", *Methods in Ecology and Evolution*, 14(2). DOI: https://doi.org/10.1111/2041-210x.14049

[11] D. Strömbom, and A. J. King (2018), "Robot Collection and Transport of Objects: A Biomimetic Process", *Frontiers in Robotics and AI,* 5. DOI: https://doi.org/10.3389/frobt.2018.00048

[12] R. Vaughan, N. Sumpter, A. Frost, and S. Cameron (1998)"Robot Sheepdog Project achieves automatic flock control", *Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior on From Animals to Animats* 5. DOI: https://doi.org/10.7551/mitpress/3119.001.0001

[13] A. A. Paranjape, S. -J. Chung, K. Kim and D. H. Shim, "Robotic Herding of a Flock of Birds Using an Unmanned Aerial Vehicle," in *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 901-915, DOI: https://doi.org/10.1109/TRO.2018.2853610

[14] H. T. Nguyen , T. D. Nguyen, M. Garratt, K. E. Kasmarik, S. Annavatti, M. Barlow, and H. A. Abbas, (2019) " A Deep Hierarchical Reinforcement Learner for Aerial Shepherding of Ground Swarms", *International Conference on Neural Information Processing*, vol 11953. Springer, Cham. DOI: https://doi.org/10.1007/978-3-030-36708-4_54

[15] H. T. Nguyen , T. D. Nguyen, V. Tran, M. Garratt, K. E. Kasmarik, S. Annavatti, M. Barlow, and H. A. Abbas, (2020) " Continuous Deep Hierarchical Reinforcement Learning for Ground-Air Swarm Shepherding", *arXiv*, DOI: https://doi.org/10.48550/arXiv.2004.11543

[16] X. Zhao, F. Pu, Z. Wang, H. Chen and Z. Xu (2019), "Detection, Tracking, and Geolocation of Moving Vehicle From UAV Using Monocular Camera," in *IEEE Access*, vol. 7, pp. 101160-101170, DOI: https://doi.org/10.1109/ACCESS.2019.2929760

[17] P. Kachroo, S. A. Shedied and H. Vanlandingham (2002), "Pursuit evasion: the herding noncooperative dynamic game - the stochastic model," in *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 32, no. 1, pp. 37-42, DOI: https://doi.org/10.1109/TSMCC.2002.1009131

[18] T. Miki and T. Nakamura (2006), "An Effective Simple Shepherding Algorithm Suitable for Implementation to a Multi-Mmobile Robot System," *First International Conference on Innovative Computing, Information and Control* - Volume I (ICICIC'06), Beijing, China, pp. 161-165 DOI: https://doi.org/10.1109/ICICIC.2006.411

[19] H. Hoshi, I. Iimura, S. Nakayama, Y. Moriyama and K. Ishibashi, (2018). "Robustness of Herding Algorithm with a Single Shepherd Regarding Agents' Moving Speeds". *Journal of Signal Processing*. 22. 327-335. DOI: https://doi.org/10.2299/jsp.22.327

[20] H. Hoshi, I. Iimura, S. Nakayama, Y. Moriyama and K. Ishibashi (2018), "Computer Simulation Based Robustness Comparison Regarding Agents' Moving-Speeds in Two- and Three-Dimensional Herding Algorithms," *Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)*, Toyama, Japan, pp. 1307-1314 DOI: https://doi.org/10.1109/SCIS-ISIS.2018.00205

[21] Lee W, Kim D (2017), "Autonomous Shepherding Behaviors of Multiple Target Steering Robots", *Sensors (Basel)*, 25;17(12):2729. DOI: https://doi.org/10.3390/s17122729

[22] Y. Tsunoda, Y. Sueoka, Y. Sato and K. Osuka (2018), "Analysis of local-camera-based shepherding navigation", *Advanced Robotics*, 32:23, 1217-1228, DOI: https://doi.org/10.1080/01691864.2018.1539410

[23] Y. Tsunoda, Y. Sueoka and K. Osuka (2017), "On statistical analysis for shepherd guidance system," in *IEEE International Conference on Robotics and Biomimetics (ROBIO), Macau, Macao*, pp. 1246-1251 DOI: http://doi.org/10.1109/ROBIO.2017.8324588

[24] K. Fujioka and S. Hayashi (2016), "Effective shepherding behaviours using multi-agent systems," *IEEE Region 10 Conference (TENCON)*, Singapore, pp. 3179-3182 DOI: https://doi.org/10.1109/TENCON.2016.7848636

[25] O. Burchan Bayazit, Jyh-Ming Lien and N. M. Amato (2002), "Roadmap-based flocking for complex environments," *10th Pacific Conference on Computer Graphics and Applications Proceedings.*, Beijing, China, pp. 104-113 DOI: https://doi.org/10.1109/PCCGA.2002.1167844

[26] Jyh-Ming Lien, S. Rodriguez, J. Malric and N. M. Amato (2005), "Shepherding Behaviors with Multiple Shepherds," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, pp. 3402-3407, DOI: https://doi.org/10.1109/ROBOT.2005.1570636

[27] S. Razali, Q. Meng and S. -H. Yang (2010), "A refined immune systems inspired model for multi-robot shepherding," *Second World Congress on Nature and Biologically Inspired Computing (NaBIC)"*, Kitakyushu, Japan, pp. 473-478, DOI: https://doi.org/10.1109/NABIC.2010.5716358

[28] M. Bacon and N. Olgac (2012), "Swarm herding using a region holding sliding mode controller" *Journal of Vibration and Control* ;18(7):1056-1066. DOI: https://doi.org/10.1177/1077546311411346

[29] A. Pierson and M. Schwager, "Controlling Noncooperative Herds with Robotic Herders," in *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 517-525, DOI: https://doi.org/10.1109/TRO.2017.2776308

[30] S.A Shedied (2013), "Optimal trajectory planning for the herding problem: a continuous time model". *Int. J. Mach. Learn. & Cyber.* 4, 25–30. DOI: https://doi.org/10.1007/s13042-012-0071-2

[31] J.F. Harrison, C. Vo, J. M. Lien (2010), "Scalable and Robust Shepherding via Deformable Shapes". In *Motion in Games*. MIG 2010. Lecture Notes in Computer Science, vol 6459. Springer, Berlin, Heidelberg. DOI: https://doi.org/10.1007/978-3-642-16958-8_21

[32] J. Terven, D. Cordova-Esparza (2023), "A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond". *arXiv preprint*. DOI: https://doi.org/10.48550/arXiv.2304.00501