Bio-Inspired Distributed Robotics

Sheep-Like Collective Behavior using Distributed Algorithms

Edoardo Barba

***Abstract*—In recent years, the fusion of bio-inspired techniques and distributed systems has opened new possibilities in robotics. This project delves into the utility of combining distributed algorithms with sheep-like behaviors to enhance collective behavior in robots.**

In this report, I present my findings, methodologies, and experimental results in two scenarios: one with a stationary target and one with a dynamic target. I employ distributed weighted least squares (WLS) and distributed Kalman filtering (KF) techniques to enable robot collaboration and perception.

***Index Terms*—Distributed Systems, WLS, KF, Sheep Herd**

I. Introduction

In this project, my primary objective is to use distributed algorithms and explore their application with a group of robots that behave like sheep in reaching a target. The collective behavior is inspired by the alternating leader-follower pattern observed in sheep, as elucidated in the paper "Intermittent collective motion in sheep results from alternating the role of leader and follower" by Gómez-Nava, et al. [2]. This report thoroughly explains how I approach this, detailing simulations and the experimental outcomes.

II. Problem Formulation and Model Description

This project employs a 2D Cartesian space to represent robots as simple points, enabling unrestricted movement within the simulation. Two distinct simulation types were created, each with its specific duration, where each simulation step corresponds to one second.

The simulated space features a target area represented by a circle. The target can be either fixed or dynamic, as detailed in Section VI.

A. Robot Model

The robots can move freely in their environment without any constraints. They are capable of navigating in any direction and following different paths within their workspace without limitations. I will explain their specific motion equation in the next section.

Each robot is equipped with two essential sensors:

- **GPS Sensor**: This sensor provides each robot with an estimate of its own position, albeit with a certain degree of uncertainty. To manage this uncertainty I use a scaling factor, $scale\_GPS$, which allows to control the level of imprecision. The $scale\_GPS$ factor is multiplied with the covariance matrix and allows for an increase or decrease in the uncertainty.
  I then generate a covariance matrix by randomizing a 2x2 matrix with values ranging from -0.5 to 0.5. By performing a matrix multiplication with its transpose, we derive the final covariance matrix, R_GPS, which quantifies the uncertainty associated with the GPS measurements.

- **Target Position Sensor**: This sensor gives the robot an estimation of the target's position at each time step. It supplies coordinates in the robot's local Cartesian plane, which are later converted into global coordinates.
  Similar to the GPS sensor, this sensor introduces a degree of uncertainty, which is quantified using covariance matrix R, multiplied by a scaling factor $scale\_R$.

Apart from utilizing GPS measurements to update its own position, the robot also employs an estimate of its input control, also characterized by some uncertainty, to compute its new position.

The robot employs a Kalman filter algorithm, wherein the prediction step, it utilizes the estimated input control to project and anticipate the new position. It then updates the position estimate by integrating the GPS data.

III. Sheep behaviour

I simulated the robots to behave like sheep, inspired by the paper [2]. In this paper, Gómez-Nava, et al. studied the spontaneous behavior of small sheep flocks and found that sheep exhibit a collective behavior that consists of a series of **Collective Motion Phases** (CMP) interrupted by **Grazing phases** (GP). Each motion episode has a temporal leader that guides the group in line formation. They provide evidence that group coordination in these episodes results from the propagation of positional information of the temporal leader to all group members through a strongly hierarchical, directed interaction network. Furthermore, they show that group members alternate the role of leader and follower by a random process, which is independent of the navigation mechanism that regulates collective motion episodes.

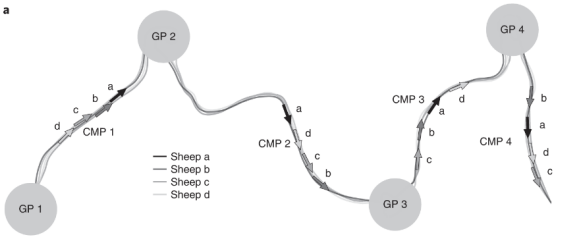


Fig. 1: Visualization of sheep behavior with GP and CMP phases

A. Intermitted Collection Motion: CMP and GP

As anticipated, the group behaves with a series of spontaneous collective motion episodes (CMPs) that are preceded and followed by a Grazing Phases (GPs). In a GP, individuals spend most of the time with the heads down and grazing. In a CMP, instead, individuals move together and do it in line formation. CMPs often involve large collective displacements, with the group performing several turns while remaining cohesive. The order of the individuals in line formation remains, in the vast majority of the CMPs, unchanged. Starting with all the individuals in a GP, a CMP begins when one group member decides to move and its motion elicits a collective displacement. On the other hand, a CMP typically ends when the individual at the front of the moving group stops moving. Collective transitions GP $\rightarrow$ CMP and CMP $\rightarrow$ GP are very fast processes, occurring in a couple of seconds.

Each GP phase lasts 30 seconds (30 time steps), while the CMPs last 20 seconds. In both phases, at each step, each one measures the position of the target and shares information with other robots, using a distributed algorithm to estimate the position of the target, as explained in section V and a local Kalman filter to estimate their own position. At each time step, each robot does 3 things: move (GP or CMP), measure, and share.

1) Grazing Phase: During this phase, they have, for the majority of the time, velocity 0, while randomly they slightly move returning to being stationary afterward. During GP, each robot, at each step has a certain probability of becoming the leader and making the Collective Motion Phase beginning. This probability increases proportionally to the number of steps within the GP.

2) Collective Motion Phase: When a robot assumes the role of the leader, a rank assignment procedure is initiated. The leader is assigned a rank of 0, while the remaining robots are assigned random ranks. These ranks dictate their positions within the formation, establishing a distinct order in which they move. The leader uses the local information he has about the position of the target to move towards its position. Information flows from the individual at the front to all group members via a directed one-dimensional network in which the individual in (K+1)$^{\text{th}}$ position in the file receives the information from the individual in the K$^{\text{th}}$ position. Hence, the Interaction Network implemented consists of a system in which each robot follows only one other robot, based on the rank they have.

This information flow resembles the model 3 described in the paper [2].

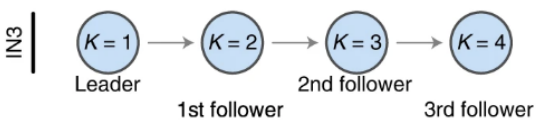


Fig. 2: Representation of information pooling from leader to other robots

IV. Equations of Motion

The equations of motion can be described as follows:

$$\dot{x}_i(t) = v(t)e(\theta_i) \tag{1}$$

$$\dot{\theta}_i(t) = \sum_{j=1}^{N} A_{ij}(t)g_{ij}(x_i, \theta_i, x_j, \theta_j) + 2\sqrt{D_\theta}\xi_i(t) \tag{2}$$

Where $\hat{e}(\cdot) := \cos(\cdot)\hat{x} + \sin(\cdot)\hat{y}$, θ_i encodes its moving direction, and $v(t)$ is a stochastic (global) variable that sets the speed of individuals ($v(t) = 0$ during GP and $v_0 \approx 1$ m/s during CMP.

In equation (2), $A_{ij}(t)$ is the element of the interaction network (IN) that defines the influence of individual j on i at time t, g_{ij} is the guiding mechanism (specified below), $\xi_i(t)$ is a delta-correlated noise, and D_θ is the noise strength.
During GPs, all $A_{ij} = 0$, and as the group transitions from GP to CMP, after the rank assignment, the values of A_{ij} become as follows:

$$A = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{3}$$

Where a cell A_{ij} represents the influence of an individual of rank j on an individual of rank i. The matrix of the interaction network is considered constant between each CMP.

The guiding mechanism, denoted as g_{ij}, plays a crucial role in enabling one robot to follow another. I implement a combination of an *attraction mechanism* (model P in [2]) and a *velocity alignment mechanism* (model V). The two mechanisms are weighted as follows:

$$g_{ij}(x_i, \theta_i, x_j, \theta_j) = 0.7 \cdot G1_{ij} + 0.3 \cdot G2_{ij} \tag{4}$$

$$G1_{ij} = \sin(\alpha_{ij} - \theta_i) \tag{5}$$

$$G2_{ij} = \sin(\theta_j - \theta_i) \tag{6}$$

Where $G1_{ij}$ is the attraction mechanism, and $G2_{ij}$ is the velocity alignment mechanism. While α_{ij} is the angle in polar coordinates of $\mathbf{e}_{ij} = \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} = \hat{e}(\alpha_{ij})$.

The integration of these two mechanisms enables the robots to establish a linear formation, seamlessly following one another during the CMP phase.

V. Distributed Estimation

As previously mentioned, the robots operate using a distributed algorithm, without the presence of a centralized supervisory controller. Each robot independently generates its own estimate and shares this information with other robots. In general i consider that each robot can exchange information with all other robots, but it could be decided to vary the number of connections, as I did in Section VII-B.

I have implemented two distinct simulations, which will be elaborated upon in the upcoming section VI. In the first

simulation, where the target remains stationary, a distributed Weighted Least Square algorithm has been employed. In the second simulation, instead, the target exhibits dynamic movement, and I opted for the utilization of a distributed Kalman Filter.

The robots have connections with each other, connections are bidirectional and symmetric and they are independent of the robots' distances.

A. Distributed WLS

Each robot has its own local estimate. The robot considers a local composite information matrix $F_i(k)$ and a local composite information state $a_i(k)$. The local estimates are updated at each step using the information of the other robots, with the following update rule based on local consensus:

$$F_i(k+1) = F_i(k) + \sum_{j=1}^{n} q_{ij}(k)(F_j(k) - F_i(k)) \tag{7}$$

$$a_i(k+1) = a_i(k) + \sum_{j=1}^{n} q_{ij}(k)(a_j(k) - a_i(k)) \tag{8}$$

In which the term q_{ij} represents a time varying weighting scheme. In my project I opted for maximum degree-weight, and so I chose q_{ij} as follows:

$$q_{ij}(k) = \begin{cases} \frac{1}{n} & \text{if } (j,i) \in Edges \text{ and } i \neq j, \\ 1 - \frac{di(k)}{n} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Information sharing occurs through a specified number of message exchanges during each step, and for all experiments, this number has been set to 5.

Hence at each step, each robot can compute a locally unbiased weighted least-square estimate, providing that $F_i(t)$ is invertible, with the following equation:

$$p\_est\_distr = Fi^{-1} \cdot ai \tag{10}$$

B. Distributed Kalman Filter

In the implementation of the distributed Kalman filter, the challenge is similar to the Weighted Least Squares (WLS) scenario, but it comes with an extra layer of complexity. In this case, the target area is in motion, which adds dynamic elements that we need to take into account. Each robot uses this motion to its advantage by making predictions about where the target will be next, building on what it knows from the past.

This prediction step can be expressed through the following equations:

$$p_{\text{est\_distr}} = A_t \cdot p_{\text{est\_distr}} + Bt \cdot u\_target \tag{11}$$

In this equation, A_t and B_t represent the transition matrix and control matrix, which contain the robot's knowledge about the kinematic model of the target. These matrices play a key role in estimating the target's next position. While u_target represents an estimate the robot has about the velocity of the target. This estimate is constant across all the simulations.

Equation 12, instead, updates the uncertainty that the robot has regarding the target's position, which is represented by matrix Th_{KF}.

$$Th_{KF} = At \cdot Th_{KF} \cdot At^T + Qt \tag{12}$$

These operations enable the robot to enhance its understanding and update its knowledge about the target's position. To facilitate this update, the matrix Q_t is utilized, representing the level of uncertainty associated with the target's motion.

Similarly to the previous case, each robot computes the values of $a(k+1)$ and $F(k+1)$ through an average consensus process. The estimation of the target's position and the uncertainty in this estimate are determined using the following equations:

$$x_i(k+1) = P_i(k+1)\left(P_i(k+1)^- x_i(k+1) + na_i(k+1,q)\right) \tag{13}$$

$$P_i(k+1) = \left(P_i(k+1)^- + nF_i(q)\right)^{-1} \tag{14}$$

The equations make it clear that each sensor node must know the total number of nodes, represented as n, to update the estimates.

This method allows the robots to easily share information, working together to arrive at a collective estimate of the target's position in a distributed way.

VI. Implementation Details

A. Methodologies

The simulation is implemented in Python, making use of the Matplotlib framework for graph generation and NumPy for efficient data management.

There are three main classes:

1) HerdModel: The `HerdModel` class acts as the central control unit responsible for managing the entire robotic herd. This class maintains records of each individual robot and coordinates their actions. It plays a vital role in ensuring coordinated behavior within the robot team by overseeing their movements, measurements, and information sharing.

2) RobotSheep: The `RobotSheep` class represents an individual robot within the herd. It encapsulates all the local information essential for the simulation, including its estimated position and sensor uncertainty. Additionally, this class contains methods for robot movement, making measurements, and sharing information. When a robot shares its knowledge, it passes this information to the `HerdModel` class, which then disseminates it to other robots as needed.

3) Target: The `Target` class represents the objective that the sheep aim to reach. It's visualized as a circular area with a defined position in Cartesian space. The representation in space is as a circle with a 5-meter radius. Depending on the simulation type, the target can either remain stationary or move. In the case of a moving target, it has its own dynamics, and at each time step, its position is updated as follows:

$$\text{pos} = A_t \cdot \text{pos} + B_t \cdot \text{control\_input}$$

Here, both A_t and B_t are 2x2 identity matrices, and *control_input* denotes the velocity. The target maintains a constant velocity of 0.3 m/s on the x-axis, while on the y-axis, it randomly selects a velocity at each time step from a Gaussian distribution with a mean of 0 and a variance of 0.5.

B. Simulations settings

The simulation takes place in a 2D cartesian space, with a width of 120 meters and a height of 60 meters, a frame of a simulation is shown in Fig. 3.

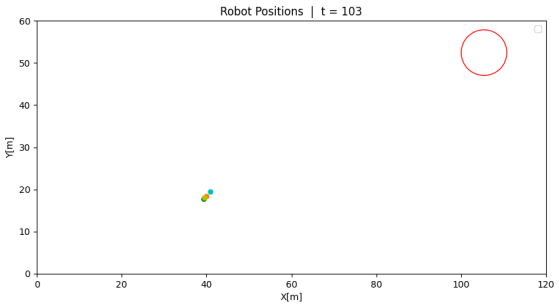


Fig. 3: Example of simulation

As anticipated previously, 2 types of simulations were implemented: a simulation with *fixed target* and a simulation with *moving target*.

During both types of simulations, the robots start in Grazing Phase and after 30 seconds begin to move with the transition to Collective Motion Phase, and so on, as explained in section III

1) simulation with fixed target: In this type of simulation, the target doesn't move. It has a fixed position in the cartesian space, which is decided randomly before the start of the simulation. The position could be between 100-110 on the x-axis and between 5-55 on the y-axis. The simulation ends when a robot reaches the target.

2) simulation with moving target: In this simulation, the initial position of the target is set at [10, 30], and it follows the dynamics described in section VI. The simulation is conducted over a duration of 300 seconds.

VII. Results

This section provides a comprehensive overview of a variety of experiments conducted to analyze the impacts of different parameters. In Table IV, you can see the parameter values that remained constant across the simulations. In all the experiments, if not specified, I kept the $scale_R$ value the same for all the robots.

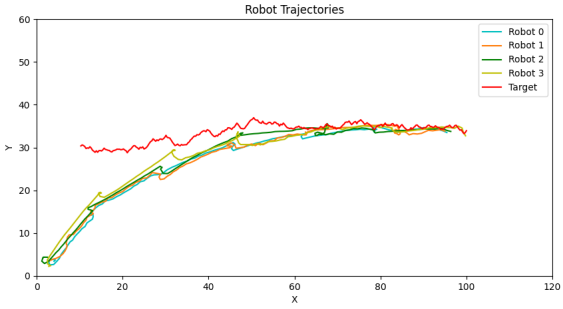
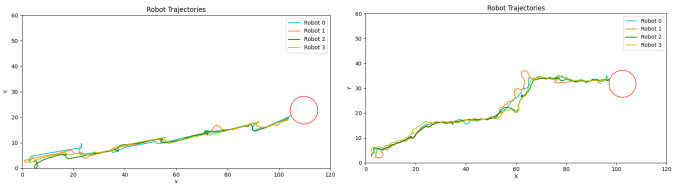


Fig. 4: Example of trajectories of robots with moving target

A. Experiment 0

In this preliminary experiment, I demonstrated that, with fixed target, an increase in the uncertainty of the target position sensor leads to a corresponding increase in the average time required for robots to reach a fixed target. This experiment involved four robots, each having the maximum number of connections. The results are summarized in Table III in the Appendix.

In Figure 5, the trajectories of the robots are visualized. The image on the left corresponds to a scenario where $scale_R$ is set to 1, while the image on the right depicts a situation with $scale_R$ set to 100. Notably, with a lower $scale_R$, which implies greater sensor precision, the robots efficiently reach the target's position more rapidly, taking a direct and precise route. In contrast, when dealing with substantial uncertainty, the robots tend to reach the target via a circuitous path during the CMP phase.



(a) scale_R=1 (b) scale_R=100

Fig. 5: Robot trajectories with different target sensor uncertainty

This happens because each robot acting as a leader, benefits from a more accurate understanding of the target's direction. As a result, they can follow a more straightforward path to reach the target, leading to faster convergence.

B. Experiment 1

In the second experiment, I examined the impact of the number of connections on the average time it takes for robots to reach a fixed target. Specifically, I used a setup with four robots and varied the number of connections among them, considering four different scenarios: 0 connections, 2 connections, 4 connections, and 6 connections.

Additionally, I introduced variations in the uncertainty of the target position sensor by adjusting the scale factor, denoted as "$scale\_R$." I tested the impact of three different scale values: 1, 10, and 100.

As depicted in Figure 6, a higher number of connections between robots leads to a reduction in the average time required to reach the target. This phenomenon can be attributed to the increased capacity for information exchange among the robots, which, in turn, allows for a more precise estimation of the target's position.

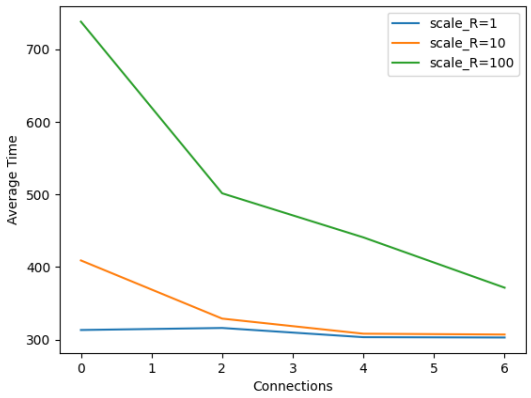


Fig. 6: Relation between number of connections and average time to reach target

By increasing the number of connections, we effectively leverage the collective measurement capabilities of a greater number of independent sensors. Consequently, it enhances the overall information available to the system.

It's worth noting from the plot that the significance of communication becomes more apparent as the scale of uncertainty ($scale_R$) increases.

More precisely, in cases where R is small, resulting in highly accurate sensors, the gradual introduction of connections has a relatively minor impact. This occurs because each robot, working autonomously, can already attain a reasonably precise estimation.

Conversely, as R grows larger, communication assumes an increasingly important role in refining the target's localization.

C. Experiment 2

In the third experiment, I focused on a dynamic target and investigated the influence of the prediction step within the Kalman filter on the group's ability to track the target. The objective is to see how the accuracy of the prediction step affects the group's ability to effectively track the moving target. To measure this, I considered the average distance of the robots from the target throughout the simulation as a metric.

I designed two distinct scenarios for this experiment.

In the first scenario, each robot has the ability to make highly accurate predictions regarding the new position of the target. Specifically, they predicted the x-coordinate of the target's position with zero uncertainty, thanks to their precise knowledge of the target's x-velocity. At the same time, they estimated the y-coordinate with an uncertainty of approximately ±0.5 meters.

In the second scenario, I introduced a different condition where the robots' prediction capabilities were less precise.

Table I presents the results obtained in the experiment, displaying the mean distance of robots from the target averaged across all simulation runs.

TABLE I: Experiment 3 Results

Scale_R	**High Accuracy**	**Low Accuracy**
10	18.09 m	17.89 m
100	17.53 m	27.95 m
1000	21.56 m	45.31 m

As evident from the table, when prediction accuracy is higher, robots generally exhibit a reduced average distance during simulations, meaning that they tend to follow better the target area.

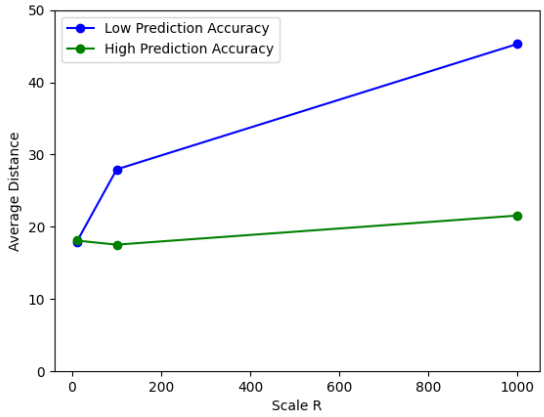


Fig. 7: Average distance with different prediction precision

In image 7 the same results are visualized in a plot.

It's worth noting that, with a high level of precision in prediction, the fluctuations in the uncertainty of the target position sensor don't have a significant impact. Even in scenarios with very high sensor uncertainty ($scale_R$ equal to 100 or 1000), the robots maintain their ability to effectively track the target. This highlights that variations in sensor uncertainty play a relatively minor role when the prediction step is very accurate.

D. Experiment 3

In this last experiment, I explore the importance of alternating the role of leader during CMP phases. I considered a fixed target and 4 robots. I consider $scale_R$ not constant across all robots but one robot with $scale_R = 10$ and the other 3 with $scale_R = 1000$. So one robot has a precise sensor while the others have a very high noise.

As previous experiments, every time the CMP phase begins, the leader is chosen completely randomly. In figure 8 we can see the trajectories of the robot in a simulation with this setting. Robot number 0 is the one with $scale_R = 10$, so it is the only one with an accurate estimate.

As can be seen from the image, the robots manage to converge to the target position. When robot 0 leads, the group gets closer to the correct target position. But when the others take turns as leaders, they steer the group in less accurate directions.

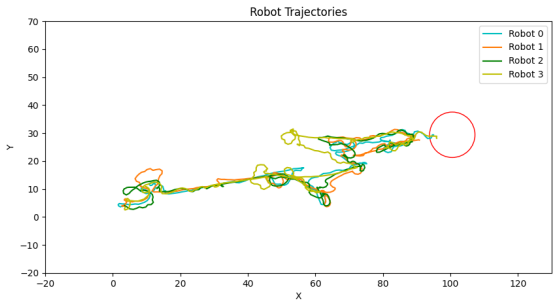


Fig. 8: Trajectories of robots

	Alternate Role	Constant Role
Average Time	1487.0	1069.9
Standard Deviation	782.09	353.59

TABLE II: Comparison of different choices of leader selection

I tried to explore the effects of alternating leadership roles between Collective Motion Phases (CMP) compared with maintaining a constant leader throughout the simulation. The results of 10 simulation runs, each with a maximum step limit of 2000, are summarized in Table II.

The data indicates that when leadership roles are alternated, the average time required to reach the target is notably shorter, and the standard deviation is reduced. This outcome can be attributed to the fact that, in simulations where leadership remains constant, if the leader possesses an accurate sensor, the group reaches the target rapidly (around 300 seconds). Conversely, when one of the other three robots assumes the role of leader, they may struggle to guide the group to the target destination, potentially resulting in significantly longer times or even failed convergence.

This highlights the importance of a temporal-leader-selection mechanism in order to exploit the information of every group member. In particular, the random leader selection policy used in these experiments, where all group members have an equal chance of becoming temporary leaders (a strategy known as democratic leadership in social sciences [1]), enables individuals to share information effectively, even without information exchange, and even when they are unaware of which group members possess crucial information. [2]

VIII. Conclusion

In this project, I investigated the application of distributed algorithms within a group of robots emulating sheep-like behavior to reach a target area. Two distributed algorithms were implemented to facilitate information exchange among the robots, enabling them to reach a consensus on the target's position.

I explored various scenarios, delving into the impact of different parameters on the simulation's outcomes. Particularly, I focused on the significance of the number of connections between robots, highlighting the critical role of communication, especially when sensor uncertainty is high.

Experiment 2 involved an examination of the influence of the prediction step within the Kalman filter, demonstrating how high prediction precision can significantly enhance the algorithm's efficiency.

Lastly, I delved into the importance of alternating the leader's role when message exchange is not employed. This approach fosters information sharing without revealing which individuals possess crucial data.

It's important to note that this strategy is implemented to achieve the goal across multiple Collective Motion Phases (CMPs), with information aggregation taking the form of a temporal average.

This project has achieved distributed collective behavior in a group of robots, demonstrating robustness to sensor uncertainty when complemented by effective communication. By exploring the advantages of alternating leadership roles during Collective Motion Phases (CMPs), this work underlines bio-inspired potential strategies for enhancing group navigation, by mirroring behaviors observed in various social animal systems

REFERENCES

[1] John Gastil. A definition and illustration of democratic leadership. *Human Relations - HUM RELAT*, 47:953–975, 08 1994.

[2] Luis Gómez-Nava, Richard Bon, and Fernando Peruani. Intermittent collective motion in sheep results from alternating the role of leader and follower. *Nature Physics*, 18:1–8, 10 2022.

APPENDIX A
ADDITIONAL GRAPHS

TABLE III: Experiment 0 Results

n_connections	scale_R	Average Time (s)
0	1	313.25
2	1	316.05
4	1	303.35
6	1	302.90
0	10	409.10
2	10	328.95
4	10	308.15
6	10	307.00
0	100	738.10
2	100	501.60
4	100	440.85
6	100	371.55

TABLE IV: Experiments Parameters

Parameter Name	Description	Value/Range
Sensor covariance (Ri)	Covariance matrix for sensor measurements	Random between -0.5 and 0.5, scaled with ”scale_R”
Sensor Model (Hi)	Sensor model matrix for measurement noise	Random between -0.5 and 0.5
Input Uncertainty (Qi)	Covariance matrix for noise on robot input control	Random between -0.5 and 0.5, scaled by 1
GPS Uncertainty (R_GPS)	Covariance matrix for GPS noise	Random between -0.5 and 0.5, scaled with ”scale_GPS”
Robot positions	Initial random position for the robots	Random between 2-5 on both x and y
Target position	Initial random position for the target	[10, 30] for fixed target, or random between 100-110 on x and 5-55 on y