Bio-Inspired Swarm Flocking with Minimal Communication for Adaptive Search Operations

Venkata Madhav Tadavarthi
M.Eng. Robotics
University of Maryland, College Park
vmadhav@umd.edu

***Abstract*—Marine robotic swarms have the potential to revolutionize deep-sea exploration and search-and-rescue (SAR) operations through scalable and efficient search strategies. However, conventional multi-robot systems often depend on acoustic communication, which is costly, energy-intensive, and unreliable in marine environments. This paper proposes a bio-inspired, decentralized swarm control framework that mimics fish-schooling behavior to enable coordination with minimal communication. The methodology integrates a modified Cohesion-Separation-Alignment (CSA) model with information-driven bias, lightweight formation control to maintain inter-agent spacing, and a behavior-based finite state machine for search optimization. A Python simulation framework is used to model marine dynamics and evaluate swarm performance. To reduce communication overhead, an event-triggered mechanism allows robots to operate autonomously, sharing information only when objects of interest are detected.**

***Index Terms*—Underwater robotics, swarm intelligence, Cohesion-Separation-Alignment (CSA) model, bio-inspired flocking, decentralized control, minimal communication, Finite State Machine (FSM), Search and Rescue (SAR).**

I. Introduction

Underwater robots are crucial for deep-sea exploration and search & rescue (SAR) missions. However, traditional single-robot systems are often inefficient due to limited coverage, high energy consumption, and long mission durations. Swarming multiple robots offers a more scalable and efficient solution, though it comes with challenges, particularly the heavy reliance on acoustic communication, which is costly, unreliable, and energy-draining. The underwater environment also presents obstacles such as low visibility, bio-life interference, and ocean currents, making conventional land-based swarm control methods ineffective. This project addresses these challenges through a decentralized control framework inspired by fish-schooling behavior, utilizing a modified Cohesion-Separation-Alignment (CSA) model biased by local information density to guide exploration. Adaptive formation control is incorporated to maintain optimal inter-agent spacing and enable self-stabilization under environmental disturbances. To minimize communication overhead, a behavior-based finite state machine (FSM) governs event-triggered transitions between exploration, investigation, and alert modes, enabling robots to act autonomously using onboard sensors. The project aims to develop and simulate an autonomous surface swarm system with bio-inspired flocking for search tasks, with adaptive formation control in a simulated ocean environment using a decentralized control system to enable minimal communication while maintaining coordination.

II. Related Work

There have been many advancements in the field of underwater robotics and swarm implementations. Zhou et al. [1] conducted a survey on underwater multi-robot systems (UMRS), discussing various projects developed with consideration of factors such as sensor collaboration, control methods, role-based approaches, and behavior techniques. Cai et al. [2] explored cooperative artificial intelligence for underwater robotic swarms, aiming to enhance the efficiency and intelligence of multiple underwater robots working together. Their work delves into AI-driven cooperative mechanisms, including formation control, task allocation, path planning, obstacle avoidance, flocking control, and cooperative communication and navigation.

Several swarm optimization algorithms have been proposed, such as Krishna et al.'s [3] review on Particle Swarm Optimization and Rendezvous Swarm Algorithms. Connor et al. [4] reviewed current algorithms, communication methods, and designs for underwater swarm robots, providing valuable insights on centralized vs. decentralized techniques as well as various swarm algorithms, including the Artificial Fish Swarm Algorithm (AFSA). Masaru et al. [5] examined the dynamics of emergent flocking behavior, discussing the Cohesion-Separation-Alignment (CSA) model, inspired by Reynold's distributed behavior model.

A few strategies were explored in the context of formation control, including A. Riah et al. [6], where formation control of a multi-robot system was achieved using virtual structures. Farhad et al.'s [7] distance-based formation control for multi-agent systems helped in designing an adaptive formation control. Benjamin et al.'s [8] approach, which focuses on information-theoretic planning, led to a modified CSA architecture for better flocking behavior.

Among the various heuristic and search optimization algorithms, Antoine et al.'s [9] work on bio-inspired heuristically accelerated reinforcement learning (RL) for underwater multi-agent behavior stood out. In addition to that, Ross et al.'s [10] SAR mission with autonomous flying robots that focuses on behavior-based cooperative intelligence has inspired the idea of an interface for the optimal search strategy.

III. Problem Statement

Given:

- A set of n autonomous surface vehicles $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ operating in a marine environment.
- Each robot a_i has:
  - Local position $p_i(t)$ and velocity $v_i(t)$,
  - A finite set of onboard sensor measurements $S_i(t)$.
- A search region containing one or more unknown targets $T = \{t_1, t_2, \ldots, t_k\}$ located at unknown positions in the environment.

Objective: Design a decentralized control policy $\pi_i : S_i(t) \rightarrow \mathbf{v}_i(t)$ for each agent a_i, such that the following are achieved:

1) *Emergent Flocking:* Maintain cohesive swarm behavior using a modified Cohesion-Separation-Alignment (CSA) model with an information-based bias as defined:

$$\vec{F}_{fi} = \vec{F}_{ci} + \vec{F}_{si} + \vec{F}_{ai} + \vec{F}_{info} + \vec{F}_{form} \tag{1}$$

2) *Formation Control:* Maintain inter-agent spacing through a formation force as defined:

$$\vec{F}^i_{form} = w_f * \sum_{j=1}^{n_i} (|\vec{p}_{ij}| - p') * \vec{e}_{p_{ij}} \tag{2}$$

3) *Search Optimization:* Minimize search time by maximizing coverage of the environment via an event-triggered behavior-based FSM with states: $S = \{\text{Search}, \text{Mission}, \text{Rescue}\}$. where transitions are based on onboard detections (sensor-inputs) and pre-defined thresholds.
4) *Minimal Communication:* Ensure coordination with minimal inter-robot communication by leveraging local sensing and only triggering messages when necessary.

Goal: Minimize the expected search time $\mathbb{E}[T_{\text{detect}}]$ to locate all targets $t_i \in T$, while satisfying swarm cohesion, formation constraints, and communication efficiency:

$$\min_{\pi_1, \ldots, \pi_N} \quad \mathbb{E}[T_{\text{detect}}] \quad \text{s.t.} \quad \|\vec{p}_{ij}\| \leq p', \quad \forall i, j. \tag{3}$$

IV. Methodology

The proposed methodology presents a decentralized swarm framework that integrates emergent fish-school-inspired flocking behavior using a modified CSA model, enhanced by lightweight formation control for maintaining optimal inter-agent spacing. An information-driven search mechanism biases the swarm toward unexplored regions, while a behavior-based finite state machine enables event-triggered switching between exploration, investigation, and alert modes.

A. Flocking Behavior

The swarm exhibits decentralized fish-schooling behavior using the Cohesion-Separation-Alignment (CSA) model [5]:

- **Cohesion**: Each agent moves toward the centroid of local neighbors, maintaining swarm integrity.
- **Separation**: Prevents collisions by maintaining an optimal inter-robot distance.
- **Alignment**: Synchronizes movement direction within the swarm for coordinated motion.

These behaviors collectively enable self-organized and distributed navigation in dynamic environments without continuous communication. Each agent is assumed to have two values, Position vector $\vec{p_{ij}}$ of agent i to the neighboring agent j which is used to calculate the central positional vector of neighboring flockmates $\vec{P_i}$ (4). Another value is the realtive velocity vector of agent i w.r.t j which is given as v_{ij} (6) that is used to calculate the average relative velocity vector V_i (6) of neighboring flockmates.

$$\vec{P_i} = \frac{1}{n_i} \sum_{j}^{n_i} \vec{p}_{ij} \tag{4}$$

$$\vec{v}_i j = \frac{d\vec{p}_{ij}}{dt} \tag{5}$$

$$\vec{V_i} = \frac{1}{n_i} \sum_{j}^{n_i} \vec{v}_{ij} \tag{6}$$

Using these assumptions, we compute the total force vectors for the CSA action rules (7) as:

$$\vec{F}_{fi} = \vec{F}_{ci} + \vec{F}_{si} + \vec{F}_{ai} \tag{7}$$

Where $\vec{F_{fi}}$ is the total flocking force vector and $\vec{F_{ci}}$ is the force vector associated with the cohesion force, $\vec{F_{si}}$ is associated with the separation force and $\vec{F_{ai}}$ is associated to alignment force. These forces can be defined as the following.

$$\vec{F}_{ci} + \vec{F}_{si} = \vec{F}_{csi} = (w_{ci} - \frac{w_s i}{|\vec{P_i}|})\vec{e}_{P_i} \tag{8}$$

$$\vec{F}_{ai} = w_{ai}\vec{e}_{V_i} \tag{9}$$

Where w_{ci}, w_{si} and w_{ai} are positive coefficients. $\vec{e}_{P_i}$ and $\vec{e}_{V_i}$ are the unit vectors associated to the central positional vector $\vec{P_i}$ and the average velocity vector $\vec{V_i}$.

For a fixed CSA, these rules do not change in response to dynamic factors, such as obstacles or other perturbations in the environment. Hence, we focus on adaptive CSA action rules, which modify the parameters or the behavior itself based on real-time environmental inputs. This helps ensure that flocking behavior is achieved even during obstacle avoidance, and prevents disjoining of the agents. This can be implemented as: $w* = f(w)$ where w is the static parameter or the weight factor and f is a function that adjust the weightings of each behavior based on current conditions. In our paper, we implement something called 'Modified CSA Action Values' by introducing an information-based bias using the gradient of an information map I(x,y) that generates a force that pushes the robot towards unexplored areas. This accounts for effective coverage planning while ensuring the flocking behavior remains stable. It is given as follows:

$$\vec{F}_{info} = -w_i * \Delta I(x, y) \tag{10}$$

This modifes the equation (7) as follows:

$$\vec{F}_{fi} = \vec{F}_{ci} + \vec{F}_{si} + \vec{F}_{ai} + \vec{F}_{info} \tag{11}$$

B. Adaptive Formation Control

A formation control force, that maintains desired distance (spacing) is introduced [7] such that it improves area of coverage and flexiblity in dynamic environments. The formation control force $\vec{F_{form}}$ is defined as follows:

$$\vec{F}^i_{form} = w_f * \sum_{j=1}^{n_i} (|\vec{p}_{ij}| - p') * \vec{e}_{p_{ij}} \tag{12}$$

where w_f is the formation parameter/weight, p' is the desired inter-agent distance, n_i includes all neighbors or those within a sensing radius and $\vec{e}_{p_{ij}}$ is the unit vector associated with $\vec{p}_{ij}$. This formation control force gently attracts or repels agents to maintain formation spacing.

Now the modified total control force is given as the following.

$$\vec{F}_{fi} = \vec{F}_{ci} + \vec{F}_{si} + \vec{F}_{ai} + \vec{F}_{info} + \vec{F}_{form} \tag{13}$$

To generalize the overall swarm behavior and combine multiple force contributions (such as directional bias, attraction to anomalies, or goal-oriented behavior), we define the total control force as [5]:

$$\vec{F}_{\text{total}} = \vec{F}_f + p\vec{F}_d = \frac{1}{n}\left(\sum_{i=1}^{n} \vec{F}_{fi} + p\sum_{i=1}^{n} \vec{F}_{di}\right) \tag{14}$$

Here, $\vec{F}_{fi}$ is the total flocking force for agent i, $\vec{F}_{di}$ is a directional or task-oriented force (e.g., towards a target), p is a tuning weight, and n is the number of agents in the swarm. This formulation allows combining emergent behavior with global mission objectives in a unified manner.

C. Search Optimization

To minimize acoustic communication overhead, a behavior-based control strategy is implemented. We implement a finite state machine (FSM) for each agent that transitions between behaviors based on sesnor inputs and internal states. The finite state machine can be defined as a 4-tuple set given as: FSM = (S, Σ, δ, s_0). Where S is a set of states, Σ is a set of actions, δ is the transition function, and s_0 is the initial state. In this approach, we define the set S with three main behavioral states for each agent. It is as follows:

- **Search Mode:** This is a default state in which the robot performs the modified CSA and ensures flocking behavior and formation control.
- **Mission Mode:** This state is triggered when the onboard sensors of a robot detect anomalies (e.g., unusual temperature, movement, or acoustic signals). The robot focuses on the area of interest, possibly reducing speed or altering formation to gather more data.

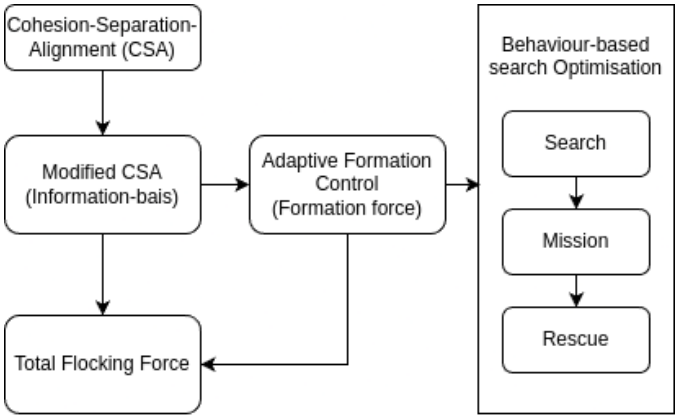


Fig. 1: Architecture

- **Rescue Mode:** Activated upon confirmation of a significant finding (e.g., a survivor). The robot sends a minimal alert to nearby robots to coordinate a response, such as convergent on the location or relaying information to a base station.

The state transition functions can be defined in such a way that they consider various anomalies, such as unusual temperatures, movement, or acoustic signals, and also for sending/receiving alerts for activating rescue mode. In Search Mode, we follow the (13) as the control force for each agent. When an anomaly is detected, the Mission Mode is activated, and we introduce a stochastic method [5] where we add a new force called $\vec{F}_{target}$ which is the vector toward the anomaly location. This ensures that flocking and formation are achieved along with an emphasis on navigating towards a given target position. The total force equation will then be modified as given in equation (15). We can reduce the alignment/formation weight to let agents deviate from the flock if needed. In Rescue Mode, we stop the robot $v_i = 0$, and the robot communicates alerts (if acoustic communication is enabled), or transmits a beacon using passive signaling (light/sound).

$$\vec{F}_{fi} = \vec{F}_{ci} + \vec{F}_{si} + \vec{F}_{ai} + \vec{F}_{target} \tag{15}$$

D. Simulation Environment

Initially, numerical simulations can be done using MATLAB/Python by verifying our control strategies and combining these control functions. This setup ensures accurate modeling of underwater locomotion, sensing, and environmental interactions while enabling decentralized multi-robot coordination.

V. Implementation

This section outlines the algorithms and pseudocode implemented throughout the project.

The development began with the implementation of the basic CSA (Cohesion-Separation-Alignment) architecture for a multi-robot system. Subsequently, a modified CSA model was introduced by incorporating an information-based bias. This enhancement led to faster response times during search and rescue missions by enabling more directed movement toward

target locations. The Algorithm for Cohesion, Separation, and Alignment forces are defined below in Algorithm 1 [11].

With the introduction of the information-bias force, we observe an improvement in the agents' performance. This is because the modified CSA algorithm enables the agents to move toward the targeted region more efficiently compared to the original method. This trade-off helps minimize the overall time required for the SAR mission. The algorithm for the information-bias force $\vec{F}_{\text{info}}$ is defined in Algorithm 2.

Algorithm 1 CSA Behavior and Goal Force

function COHESION(i, positions)
  Find neighbors within cohesion radius
  if neighbors exist **then**
    Compute center of mass of neighbors
    return vector toward center
  else
    return $[0, 0]$
  end if
end function
function SEPARATION(i, positions)
  Compute distances from agent i to all others
  Identify neighbors within separation radius
  Initialize force $\leftarrow [0, 0]$
  for all neighbor j **do**
    Add repulsive force from j to force
  end for
  return force
end function
function ALIGNMENT(i, positions, velocities)
  Find neighbors within alignment radius
  if neighbors exist **then**
    Compute average neighbor velocity
    return difference from agent i's velocity
  else
    return $[0, 0]$
  end if
end function
function GOALFORCE(position)
  Compute vector to goal
  if distance $>$ threshold **then**
    return normalized vector
  else
    return $[0, 0]$
  end if
end function

Next up, I have designed a basic finite state machine for the search optimization as shown in the figure 2. Each agent operates in one of three states: S_0 (Search), S_1 (Mission), and S_2 (Rescue). Initially, all agents begin in the Search state and navigate toward targeted regions using the information-bias force. Upon reaching a region of interest (e.g., anomaly detection or significant sensor input), the agent transitions to the Mission state, where it reduces speed and performs a focused search within that area. If a target T_i is identified, the agent switches to the Rescue state and sends an alert to nearby robots or the ground system.

Algorithm 2 Information-Bias Gradient Force

$unexplored \leftarrow 1.0 - visited$
Compute gradients: $G_x, G_y \leftarrow \text{gradient}(unexplored)$
$gx\_idx \leftarrow \lfloor positions[i, 0]/\text{cell\_size} \rfloor$
$gy\_idx \leftarrow \lfloor positions[i, 1]/\text{cell\_size} \rfloor$
if $0 \leq gx\_idx < grid\_dim$ and $0 \leq gy\_idx < grid\_dim$ **then**
  $grad \leftarrow [G_x[gy\_idx, gx\_idx], G_y[gy\_idx, gx\_idx]]$
  if $\|grad\| > 10^{-3}$ **then**
    $F_{info} \leftarrow grad/(\|grad\| + 10^{-5})$
    $acc \leftarrow acc + w\_i \cdot F_{info}$
  end if
end if

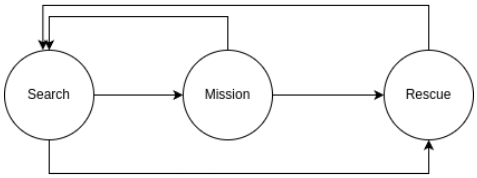


Fig. 2: Finite State Machine for Search Optimization

If no target is found after a certain duration in the Mission state (e.g., due to a false positive), the agent reverts to the Search state and resumes exploration. Similarly, after successfully completing a rescue operation, the agent transitions back to the Search state, allowing it to continue participating in the overall mission.

VI. EXPERIMENTAL RESULTS

The experimental setup was implemented in Python using a grid-based 2D environment of size 100×100, subdivided into 10×10 cells. For simplicity, it is assumed that when an agent enters a cell, its field of view (FOV) is sufficient to mark that grid as 'visited'. The simulation involves 10 autonomous agents initially positioned along the left edge of the environment (like entering the ocean), and three target points are generated randomly within the field.

To incorporate characteristics of the underwater environment, two environmental forces are introduced into the simulation: (1) a drag coefficient (16) that opposes the motion of the agents, and (2) a current flow vector (17) that disturbs the agent's trajectory. These forces are combined with the control logic to more accurately reflect real-world dynamics and improve the realism of the simulation environment.

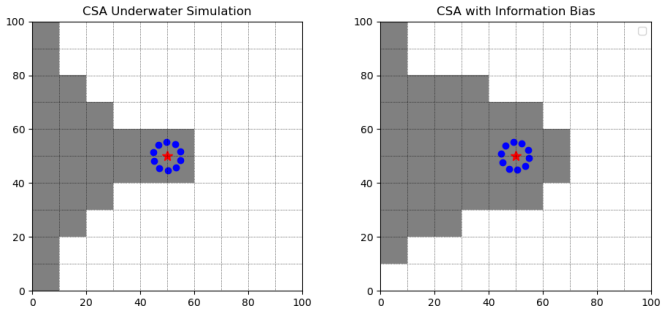
$$\mathbf{v}_{t+1} = \gamma \cdot (\mathbf{v}_t + \mathbf{a}_t + \mathbf{F}_{\text{current}}(\mathbf{p}_t, t)) \tag{16}$$

$$\mathbf{F}_{\text{current}}(\mathbf{p}, t) = \alpha \cdot \begin{bmatrix} \sin(0.1 \cdot p_y + 0.01 \cdot t) \\ 0 \end{bmatrix} \tag{17}$$

Here, α is the current strength and γ is the drag coefficient. a_t, v_t, p_t are the acceleration, velocity and position at the time t respectively.

A. *Modified CSA Results:*

The initial phase of the work focused on implementing both the standard CSA and the modified-CSA approaches, and comparing their effectiveness in the context of underwater search and rescue. In both methodologies, the objective was to guide agents toward the target regions using the control forces defined in Algorithm 1 and Algorithm 2, respectively. Upon reaching the target region, agents are expected to organize into a circular formation around the target, mimicking the biological behavior described in [5]. The resulting behaviors are illustrated in Figures 3a and 3b.



(a) CSA Output (b) Modified CSA Output

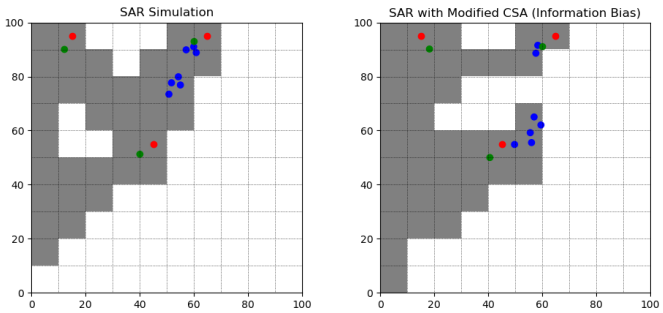
Fig. 3: Standard CSA and Modified CSA Approaches

In these images, the white grids represent *unexplored* cells, while the grey grids indicate *visited* areas. Although both outputs appear somewhat visually similar, the inclusion of the information-bias in the modified CSA causes the robots to move further forward, thereby enabling greater area exploration.

B. *Search And Rescue (SAR) Addition:*

Next, a Search and Rescue (SAR) implementation is developed to further compare the effectiveness of these two methodologies. As mentioned before, the SAR implementation consists of 10 agents and 3 randomly-generated target locations. The robots utilise this CSA/Modified-CSA flocking to navigate towards this target regions. The below figures 4a and 4b shows the results.

Each agent operates with two primary states: Search and Rescue. Agents in the Search state (shown in blue) actively explore the environment using the modified CSA algorithm. Once an agent reaches a target region, it transitions into the Rescue state (shown in green), where it halts to simulate a rescue operation. A comparative evaluation of the two methodologies is conducted in terms of coverage efficiency and agent trajectories. The results are shown in Figures 5 and 6. The Standard SAR took 436 steps whereas the SAR with Modified CSA took only 364 steps.



(a) SAR Output (b) Modified SAR Output

Fig. 4: Standard SAR and Modified SAR Approaches

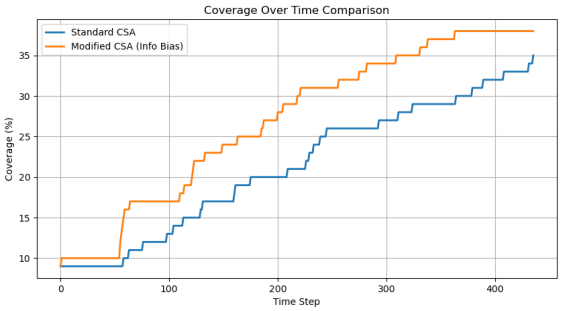


Fig. 5: Coverage Percentage vs Time

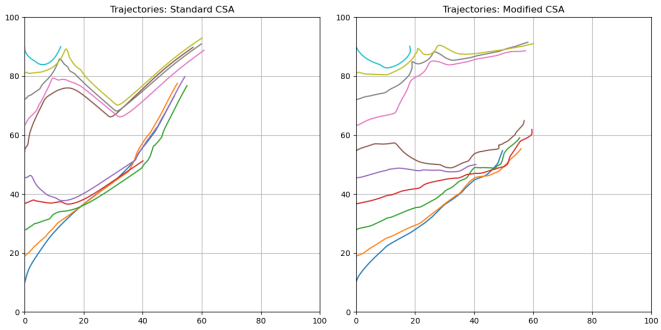
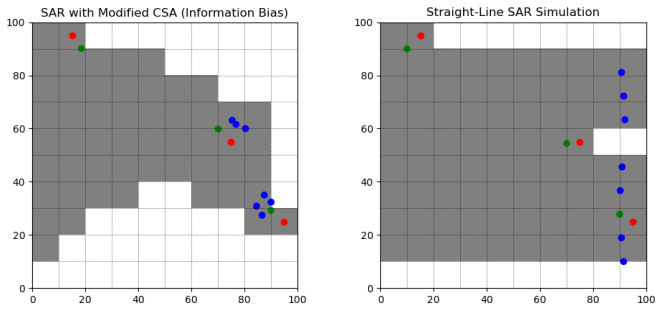


Fig. 6: Trajectories of the Agents

C. *Comparison with Baseline Approach*

To better evaluate the effectiveness of our method, I have performed the SAR mission using a baseline approach in which agents move in straight lines without coordination or adaptivity. I then compared the results with the Modified-CSA-based SAR method. The outcomes of both approaches are illustrated in Figures 7a and 7b.

Further comparison is provided in Figures 8 and 9, which show the coverage over time and the trajectories of agents, respectively. As expected, the Modified-CSA-based approach significantly outperformed the baseline strategy in terms of both efficiency and coverage. The Modified-CSA approach completed the task in 537 steps, whereas the baseline approach



(a) Modified-CSA-based SAR (b) Baseline Straight-Line SAR

Fig. 7: Comparison between Modified-CSA and Baseline SAR approaches

required 957 steps. These results clearly demonstrate the efficiency and adaptability of our proposed method.

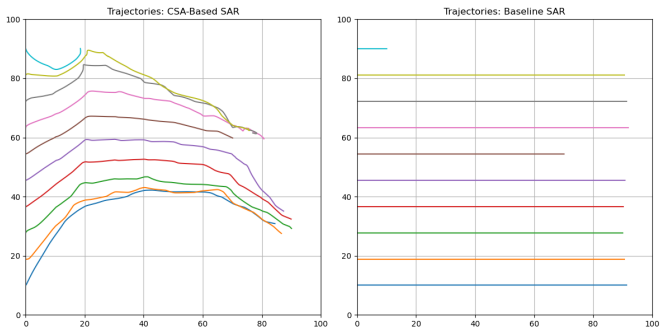


Fig. 8: Coverage Percentage vs. Time

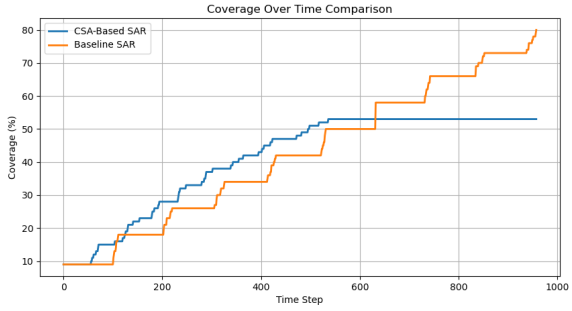
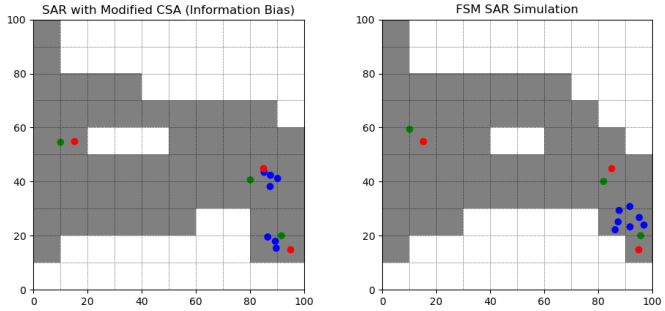


Fig. 9: Agent Trajectories for Both Approaches

D. Finite State Machine Results:

Next, I implemented a Finite State Machine (FSM) for each agent, defining three states as discussed earlier. Extending the SAR system with the modified CSA, I introduced an additional state - *Mission* which is triggered when an agent reaches a target region and begins a focused search within that area. Although this may slow down the overall search process, it is a necessary step, as the exact positions of target regions are often unknown in real-world environment. The Mission state allows agents to handle anomalies and false positives more effectively. When an agent enters Mission mode, its color changes to yellow for visualization.

Figures 11 and 12 show a comparison between the coverage and trajectories of both methodologies. As expected, the SAR with Modified CSA completed the task in 573 steps, while the FSM-based SAR took 656 steps. Although the FSM approach is slower, it prioritizes mission-awareness and communication efficiency. This results in more targeted exploration and behavior that more closely resembles realistic field operations under SAR constraints.



(a) Modified SAR Output (b) FSM Infused SAR Output

Fig. 10: Modified SAR and FSM Infused SAR Approaches

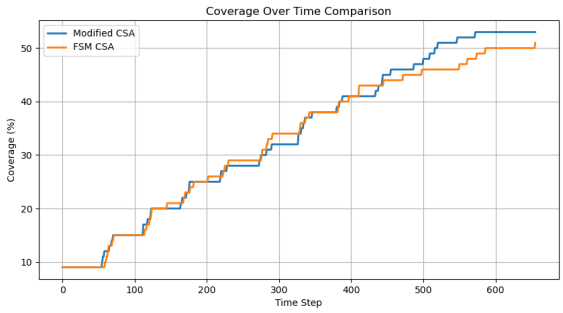


Fig. 11: Coverage Percentage vs Time

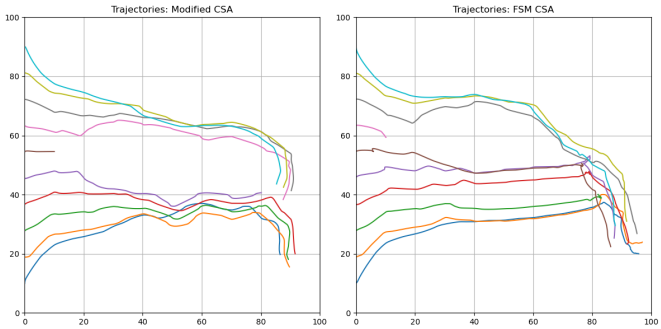


Fig. 12: Trajectories of the Agents

Comparison Results:

To validate and evaluate the proposed approach, I conducted further experiments by testing it across different swarm sizes

and varying numbers of targets. Table I and II outlines the swarm sizes, their corresponding CSA parameters, and the target counts chosen for evaluation.

TABLE I: CSA Radius Parameters for Different Swarm Sizes

No. of Agents	SEP_RADIUS	ALI_RADIUS	COH_RADIUS
5	15	12	15
10	10	8	10
15	8	7	9
20	7	6	8

TABLE II: CSA Weight Parameters for Different Swarm Sizes

No. of Agents	W_SEP	W_ALI	W_COH
5	2.0	1.2	1.5
10	1.5	1.0	1.2
15	1.3	0.9	1.1
20	1.2	0.8	1.0

Here, SEP_RADIUS, ALI_RADIUS, and COH_RADIUS denote the interaction radii used in the modified CSA (Cohesion-Separation-Alignment) model for swarm control:

- SEP_RADIUS (Separation Radius): Defines the minimum distance an agent tries to maintain from nearby agents to avoid collisions.
- ALI_RADIUS (Alignment Radius): The radius within which agents align their velocities with neighboring agents to maintain group coherence.
- COH_RADIUS (Cohesion Radius): Determines the range within which agents move toward the centroid of their neighbors to keep the swarm together.

The weight coefficients:

- W_SEP, W_ALI, and W_COH are the corresponding scalar weights applied to each behavior force (separation, alignment, and cohesion respectively). These govern how strongly each behavior influences the agent's motion.

The parameter values in Table I and Table II were selected empirically based on the density of agents in the environment. For smaller swarms (e.g., 5 agents), larger interaction radii and stronger weights were used to ensure the agents could still coordinate effectively despite being sparsely distributed. As the swarm size increased (10 to 20 agents), the radii and weights were reduced to prevent overcrowding, reduce oscillatory behavior, and ensure smoother flocking. This progressive tuning strikes a balance between local coordination and global exploration, ensuring stable and scalable behavior across varying swarm sizes.

Using these configurations, I ran the Finite State Machine–infused Search and Rescue (SAR) algorithm for different swarm sizes ($[5, 10, 15, 20]$ agents) and target counts ($[1, 3, 5]$ targets). For each combination, I recorded the final coverage percentage and the number of steps taken to complete the mission. Figures 13 and 14 illustrate the comparison of final coverage and steps taken across the different scenarios.

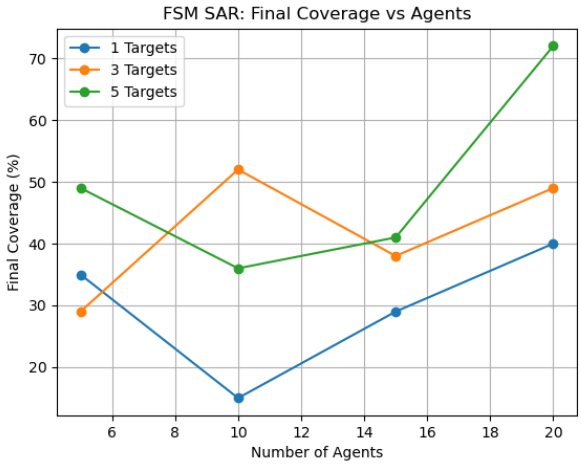


Fig. 13: Coverage Comparisons

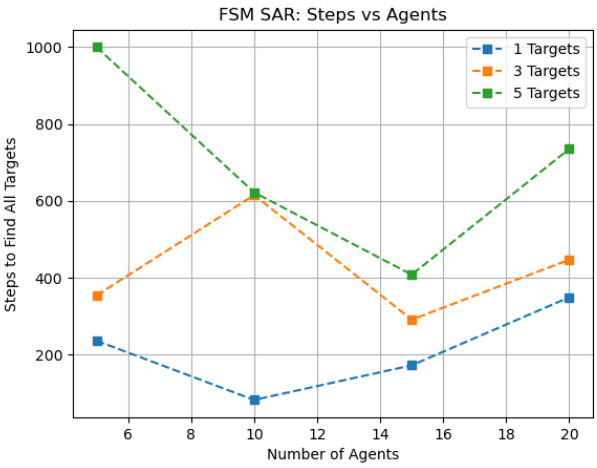


Fig. 14: Steps Comparisons

A summary of the experimental results for all combinations is provided in Table III.

TABLE III: FSM SAR Simulation Results

Agents	Targets	Coverage (%)	Steps Taken
5	1	35.00	236
	3	29.00	354
	5	49.00	1000
10	1	15.00	82
	3	52.00	615
	5	36.00	622
15	1	29.00	172
	3	38.00	291
	5	41.00	408
20	1	40.00	349
	3	49.00	447
	5	72.00	735

The complete codebase and experimental simulations for this project are available at the following repository: https://github.com/Madhav2133/enae788o-finalproj

VII. Discussions & Future Work

From Table III, we observe that for a single target, a swarm size of 10 agents yielded the best performance. When the number of targets increased to 3 and 5, a swarm size of 15 agents performed more effectively. Interestingly, when the swarm size was increased to 20 or more agents, the overall search efficiency declined slightly. This appears to be due to additional time spent on maintaining formation and flocking behavior, which can slow down target acquisition.

These results suggest that the algorithm performs optimally when the swarm size is in the range of 10 to 15 agents, balancing exploration and coordination efficiency.

It is also important to note that the parameter values used for CSA weights and radii (as shown in Table I) were chosen heuristically based on empirical understanding of the coefficient behavior. A natural direction for future work would be to develop mechanisms for adaptively tuning these parameters in real time, based on environmental feedback or mission progress, to further enhance the scalability and robustness of the system.

Furthermore, the current approach has limitations in dynamic or highly complex environments, where static assumptions about the search space may not hold. A potential extension would be to introduce dynamic obstacles into the environment, allowing the swarm to adapt to time-varying conditions and improve its robustness.

Additional directions for future work include extending the framework to 3D space, which would enable its application to Unmanned Aerial Vehicles (UAVs) and Autonomous Underwater Vehicles (AUVs). Another promising avenue is the exploration of hybrid swarms, integrating heterogeneous agents with complementary sensing and mobility capabilities—to enhance flexibility and task efficiency in diverse mission scenarios.

VIII. Conclusion

In conclusion, this project presents a decentralized, bio-inspired swarm control framework for autonomous surface vehicles, combining modified CSA-based flocking behavior with lightweight formation control to maintain inter-agent spacing. An information-biased motion strategy enhances exploration efficiency, while a behavior-based finite state machine enables event-triggered switching between search, mission, and rescue modes, reducing the need for constant inter-robot communication. Through the experimental simulations, the project evaluates the system's robustness, scalability, and energy efficiency in oceanic environments, offering a promising solution for deep-sea exploration, search-and-rescue operations, and large-scale marine monitoring.

References

[1] Z. Zhou, J. Liu and J. Yu, "A Survey of Underwater Multi-Robot Systems," in IEEE/CAA Journal of Automatica Sinica, vol. 9, no. 1, pp. 1-18, January 2022, doi: 10.1109/JAS.2021.1004269.

[2] Wenyu Cai, Ziqiang Liu, Meiyan Zhang, Chengcai Wang, Cooperative Artificial Intelligence for underwater robotic swarm, Robotics and Autonomous Systems, Volume 164, 2023, 104410, ISSN 0921-8890, https://doi.org/10.1016/j.robot.2023.104410.

[3] Ganduri Krishnavamshi, Pathri Bhargav. (2024). Swarm Intelligence in Action: Particle Swarm Optimization and Rendezvous Algorithms for Swarm Robotics. Journal of Field Robotics. 10.1002/rob.22466.

[4] J. Connor, B. Champion and M. A. Joordens, "Current Algorithms, Communication Methods and Designs for Underwater Swarm Robotics: A Review," in IEEE Sensors Journal, vol. 21, no. 1, pp. 153-169, 1 Jan.1, 2021, doi: 10.1109/JSEN.2020.3013265.

[5] Aoyagi, M., Namatame, A. (2006). Dynamics of Emergent Flocking Behavior. In: El Yacoubi, S., Chopard, B., Bandini, S. (eds) Cellular Automata. ACRI 2006. https://doi.org/10.1007/11861201

[6] Riah, Arfittariah. (2020). Formation Control of Multi-Robot using Virtual Structures with a Linear Algebra Approach. JAREE (Journal on Advanced Research in Electrical Engineering). 4. 10.12962/j25796216.v4.i1.111.

[7] Farhad Mehdifar, Charalampos P. Bechlioulis, Farzad Hashemzadeh, Mahdi Baradarannia, Prescribed performance distance-based formation control of Multi-Agent Systems, Automatica, Volume 119, 2020, 109086, ISSN 0005-1098, https://doi.org/10.1016/j.automatica.2020.109086.

[8] Charrow, Benjamin & Kahn, Gregory & Patil, Sachin & Liu, Sikang & Goldberg, Kenneth & Abbeel, Pieter & Michael, Nathan & Kumar, Vijay. (2015). Information-Theoretic Planning with Trajectory Optimization for Dense 3D Mapping. 10.15607/RSS.2015.XI.003.

[9] Antoine Vivien, Thomas Chaffre, Eva Artusi, Matthew Stephenson, Paulo Santos, et al.. Towards Bio-inspired Heuristically Accelerated Reinforcement Learning for Adaptive Underwater Multi-Agents Behaviour.

[10] Arnold, R.D., Yamaguchi, H. & Tanaka, T. Search and rescue with autonomous flying robots through behavior-based cooperative intelligence. Int J Humanitarian Action 3, 18 (2018). https://doi.org/10.1186/s41018-018-0045-4

[11] https://github.com/SverreNystad/boids-in-python.git (accessed March 2025)